N° d'Ordre :

73-14

T H E S E

PRESENTEE

A LA FACULTE DES SCIENCES
DE L'UNIVERSITE DE GRENOBLE

POUR OBTENIR

LE GRADE DE DOCTEUR-INGENIEUR
" INFORMATIQUE "

par

Gabriel ENTRESSANGLE

CO.SY.N.US., LANGAGE DE COMMANDE POUR LE SYSTEME C.A.R.I.N.E. PILOTANT DES EXPERIENCES DE DIFFRACTION NEUTRONIQUE

Soutenue le 31 janvier 1973, devant la Commission d'examen

MM.	N. GASTINEL	Président
	L. BOLLIET	Examinateurs
	R. BOUCHE	
	J.J. GIROD	

N° d'Ordre :

73-14

T H E S E

PRESENTEE

A LA FACULTE DES SCIENCES
DE L'UNIVERSITE DE GRENOBLE

POUR OBTENIR

LE TITRE DE DOCTEUR-INGENIEUR
" INFORMATIQUE "

par

Gabriel ENTRESSANGLE

CO.SY.N.US., LANGAGE DE COMMANDE POUR LE SYSTEME C.A.R.I.N.E. PILOTANT DES EXPERIENCES DE DIFFRACTION NEUTRONIQUE

Soutenue le 31 janvier 1973, devant la Commission d'examen

MM.	N. GASTINEL	Président
	L. BOLLIET	
	R. BOUCHE	Examinateurs
	J.J. GIROD	

UNIV.
LEIDEN
BIBL.

Ce travail a été réalisé au Laboratoire d'Electronique et de Technologie de l'Informatique du Centre d'Etudes Nucléaires de GRENOBLE et à l'Institut LAUE LANGEVIN.

Je suis très flatté de l'honneur que me fait Monsieur le Professeur N. GASTINEL en présidant le Jury de cette thèse et le prie de bien vouloir accepter toute ma reconnaissance.

Je tiens à exprimer toute ma gratitude à Monsieur le Professeur L. BOLLIET qui, malgré son emploi du temps chargé, a bien voulu suivre de près ce travail et me guider de ses conseils.

J'adresse mes plus vifs remerciements à Monsieur le Professeur R. BOUCHE qui a bien voulu m'honorer de sa participation au Jury.

Que Monsieur M. CORDELLE, Directeur du Laboratoire d'Electronique et de Technologie de l'Informatique, trouve ici l'expression de ma reconnaissance pour m'avoir accueilli dans son Laboratoire.

Je remercie Monsieur R. GARIOD pour l'aide et la compréhension qu'il m'a apportées au cours de la réalisation de ce travail.

Je remercie également Monsieur E. TOURNIER, chef de la Division d'Electronique Générale, qui a bien voulu m'accueillir au sein de son équipe, ainsi que Monsieur J.J. GIROD dont j'ai beaucoup apprécié les conseils éclairés.

Je suis très reconnaissant à mes camarades, Messieurs A. VINIT et P. DARIER, qui ont créé une ambiance de travail très sympathique et m'ont toujours apporté une franche collaboration.

Que toute l'équipe CARINE : Messieurs J. M. LANINO, A. BOUVIER, A. BECCIA, M. POUSSIN, M. TROTTOT, M. ROURE sache combien j'ai apprécié leur aide sympathique au cours de mon séjour parmi eux.

Je remercie toutes les personnes du Service Informatique de l'Institut LAUE LANGEVIN, en particulier Monsieur TAESCHNER, directeur, et ses collaborateurs, Messieurs KAISER, BARTHELEMY, LESOURNE, PASTOR qui ont grandement facilité ma tâche lors de mon séjour dans leurs locaux.

Je ferai une mention spéciale pour Monsieur Ph. LEDEBT, qui a guidé mes premiers balbutiements d'informaticien et m'a toujours prodigué d'utiles conseils. Qu'il soit ici publiquement remercié.

J'ai beaucoup apprécié la collaboration de Messieurs TUFFERY, LUMINEAU et OUDJAOUDI, Ingénieurs à la Société Télémécanique, qui ont été mon "fil d'ARIANE" dans le labyrinthe obscur du Superviseur et du FORTRAN temps réel.

Je remercie enfin le Commissariat à l'Energie Atomique pour la bourse d'études qu'il a bien voulu m'accorder.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41534927_0036

TABLE DES MATIERES

		Pages
INTRODUCTION		1
Partie A -	DESCRIPTION D'UNE EXPERIENCE TYPE DE DIFFRACTION NEUTRONIQUE	5
	I. Principes de la diffraction neutronique	5
	II. Montage pratique d'étude d'un cristal	6
	III. Automatisation d'une expérience de diffraction neutronique	7
	IV. Qualités requises pour un langage orienté vers la diffraction neutronique	9
	V. Solution adoptée dans le cadre du système CARINE	10
Partie B -	DESCRIPTION DU HARDWARE CONNECTE AU SYSTEME CARINE	12
	I. Le système de transmission de données CAMAC	12
	I. 1. Principe général de fonctionnement	13
	I. 2. Lignes utilisées dans l'envoi d'un ordre CAMAC	13
	I. 3. Fonctions courantes du système CAMAC	15
	I. 4. Ordre CAMAC	15
	I. 5. Description sommaire du coupleur CAMAC-T 2000	16
	II. Matériel T 2000 utilisé sur CARINE	16
	II. 1. Unité centrale	16
	II. 2. Fonctionnement des E/S	17
	II. 3. Fonctionnement des appels prioritaires	18
	II. 4. Fonctionnement des interruptions externes	19
	III. Système HARDWARE installé sur CARINE	20
	III. 1. Description de l'installation	20
	III. 2. Interruptions externes à traiter	20
	III. 3. Masquage des interruptions	22
Partie C -	ENSEMBLE DES MODULES SOFTWARE CONSTITUANT LE SYSTEME CARINE	23
	I. Contrôle de processus	23
	I. 1. Notions sur les processus	24
	I. 2. Contexte informatique pour un processus	24
	I. 3. Vue d'ensemble sur un système informatique de gestion de processus	25
	I. 4. Commandes opérateur	26
	I. 5. Action des "commandes utilisateur"	27

I.6. Modularité d'un système de contrôle de processus 28
I.7. Organisation type d'un système de commande modulaire 28
I.7.1. Module d'entrée de la commande 29
I.7.2. Module analyse 29
I.7.3. Autres modules 29
I.7.4. Schéma type d'un système de commande 30
I.8. Introduction de la multiprogrammation 30
II. Superviseur RTDMS 33
II.1.Généralités 33
II.2.Tâches de classe 1 34
II.3.Tâches de classe 2 34
II.4.Appel (ou activation) de tâches 35
II.5.Tâches swappables 37
II.6.Entrées-Sorties par MONES 37
II.7.Adressage symbolique des périphériques 38
II.8.Travaux effectués par MONES-Compte Rendu 39
II.9.Précautions à prendre dans le cas de tâches swappables 40
II.10.Système de gestion de fichier disque "SYSFIC" 40
II.11.Conclusions 41
III.Vue d'ensemble du système SOFTWARE de CARINE 43
III.1.Généralités 43
III.2.Description de l'implantation en mémoire du système 43
IV.Système de commande COSYNUS 47
IV.1.Généralités 47
IV.2.Cahier des charges primitif. Performances obtenues 49
IV.3.Traitement spécial de l'Appel Superviseur 49
IV.3.1.Généralités 49
IV.3.2.Autres problèmes liés à l'Appel Superviseur 50
IV.3.3.Réalisation pratique des sécurités 50
IV.4.Problème des entrées-sorties 50
IV.4.1.Données de départ 50
IV.4.2.Solution choisie 51
IV.5.Description d'une commande COSYNUS 51
IV.5.1.Définition 51
IV.5.2.Tâches systèmes utilisées par la commande 51
IV.5.3.Mise en place d'une nouvelle commande 52
IV.5.4.Cas spéciaux 53
IV.5.5.Règles à respecter dans l'écriture des tâches commande 53
IV.5.6.Implantation des tâches commandes 53

IV.6. Description des "Tâches Système" de COSYNUS 53
IV.6.1. Section TAAS 53
IV.6.2. Tâche TRCP 55
IV.6.3. Tâche TACLE 55
IV.6.4. Tâche ECL 56
IV.6.5. Tâche CONVER 56
IV.6.6. Tâche IMPRI 57
IV.6.7. Tâche ERREUR 57
IV.6.8. Section SPREMP 58
IV.6.9. Tâche ANALYS 58
IV.7. Commandes standard de COSYNUS 59
IV.7.1. Commande EFP 59
IV.7.2. Commande RFP 60
IV.7.3. Commande SFP 60
IV.7.4. Commande PFP 60
IV.7.5. Commande RSP 61
IV.7.6. Commande NFP 61
IV.7.7. Commande LOK 61
IV.7.8. Commande BAF 62
IV.7.9. Commande HSC 62
IV.7.10. Commande DIS 62
IV.8. Etats utilisateurs 62
IV.9. Système de gestion du Display 63
IV.9.1. Rappels sur le Display 63
IV.9.2. Modes de visualisation 65
IV.9.3. Blocs de données 65
IV.9.4. Présentation de l'image 67
IV.9.5. Organisation du système de gestion de la VISU 68
IV.9.6. Commandes VISU 68
IV.9.7. Tâches utilisées par la VISU 69
IV.9.8. Tâche INDIS 70
IV.9.9. Tâche AFFCOM 70
IV.9.10. Tâches AFFAXE, AFFVAL, COURBE, MINMAX 71
IV.9.11. Tâches EXTENS et INTEGR 71
IV.9.12. Tâche AFFDIS 71
IV.9.13. Tâche SUPDIS 72
IV.9.14. Tâche ENDVIS 72
IV.10. Sécurité de fonctionnement 73
V. Description de l'ensemble SOFTWARE chargé de la gestion du FORTRAN : "SYSFOR" 75

V.1.Généralités sur le Fortran T 2000 75
V.2.Description de SYSFOR 77
V.2.1.Tâche GESBAC 77
V.2.2.Tâche ITS 79
V.2.3.Tâche MONE 79
V.2.4.Indicateurs de SYSFOR 80
V.2.5.Tâche ASEC 81
V.2.6.Tâche SCRUTF 82
V.2.7.Tâche MODULS 82
V.2.8.Tâche INTERF 84
V.2.9.Tâche BACKGROUND 84
V.3.Fichiers FORTRAN sur disque 85
V.4.Conclusions sur le FORTRAN Temps réel 86
VI.Description de l'ensemble SOFTWARE chargé de la gestion des échanges avec le CAMAC 87
VI.1.Rappels 87
VI.2.Description globale de l'ensemble 87
VI.3.Liaisons FORTRAN-CAMAC 88
VI.4.Description du moniteur d'E/S MONAC avec le CAMAC 89
VI.5.Compatibilités Etat UC - Travail 90
VI.6.Traitement des Appels 91
VI.7.Tables de MONAC 92
VI.8.Bibliothèque CAROLUS 92
VI.9.Conclusions 94
VII. Test des modules CAMAC par FORTRAN sous le système CARINE 95
VII.1.Généralités 95
VII.1.1. Comment tester un module CAMAC 96
VII.1.2. Problèmes posés par les tests ON-LINE 96
VII.1.3. Ecriture des tests en FORTRAN 96
VII.1.4. Philosophie des tests "en ligne" 96
VII.1.5. Liste des modules pouvant être testés ON-LINE 97
VII.2. Problèmes posés par l'écriture des tests en FORTRAN 97
VII.2.1. Utilisation de la bibliothèque CAROLUS 97
VII.2.2. Découpage en temps des programmes FORTRAN 98
VII.2.3. Problèmes particuliers 98
VII.2.4. Incidents possibles - Résultats du Test 98
VII.3. Test échelle ON-LINE 99
VII.3.1. Montage Hardware nécessaire 99
VII.3.2. Paramètres à entrer 99
VII.3.3. Sous-programmes CAROLUS utilisés 99

VII.3.4. Tests effectués sur l'échelle 99
VII.4. Test ensemble Codeurs-Moteurs 103
VII.4.1. Paramètres à entrer 103
VII.4.2. Sous-programmes CAROLUS utilisés 103
VII.4.3. Tests proprement dits effectués 103
VII.5. Test du CRC 105
VII.5.1. Généralités 105
VII.5.2. Montage Hardware nécessaire 105
VII.5.3. Sous-programmes CAROLUS utilisés 105
VII.5.4. Tests effectués 105
VII.5.5. Précautions éventuelles 105
VII.6. Sous-programmes généraux communs 105
CONCLUSION 107

CO.SY.N.US., LANGAGE DE COMMANDE POUR LE SYSTEME C.A.R.I.N.E. PILOTANT DES EXPERIENCES DE DIFFRACTION NEUTRONIQUE

INTRODUCTION

Lorsque l'ordinateur est apparu pour la première fois, sa principale fonction a été de résoudre des problèmes mathématiques compliqués, qui auraient trop coûté de temps à un être humain. Cette prodigieuse rapidité dans les calculs, les tests, les décisions a donné ensuite à l'homme l'idée d'utiliser l'ordinateur dans la gestion de fichiers allant du stock de matières premières d'une usine à la population totale d'un pays. Dans ces diverses utilisations, l'ordinateur travaillait en temps différé, c'est-à-dire qu'il s'écoulait un laps de temps plus ou moins long entre la phase d'acquisition des données et l'instant de leur traitement en machine : la contrainte "temps" n'existait pas.

Alors, franchissant une nouvelle étape, l'ordinateur s'est doté d'un système de prise en compte d'appels externes, venant interrompre le cours du programme en train de s'exécuter : le "temps réel" venait de naître. Dans ce mode de travail, phase d'acquisition et phase de traitement des données sont étroitement liées, l'une pouvant interrompre l'autre, et le temps est devenu un facteur essentiel dans la conception et l'écriture des systèmes informatiques. Dans cette optique, l'ordinateur, qui était un ensemble complexe s'est transformé en petit ordinateur utilisable pour l'automatisation et la gestion de processus industriels, les gros ensembles étant de plus en plus réservés au calcul scientifique.

En fait, il serait faux de croire que le "temps réel" concurrence le "temps différé". A l'heure actuelle, le temps réel seul n'est utilisé que :

- s'il n'y a pas de calculs, mais uniquement une gestion du déroulement du processus
- si les calculs sont simples et peu coûteux en temps et en place nécessaires.

C'est donc surtout la phase d'acquisition des données qui a bénéficié des apports du temps réel, en se complétant souvent par une phase de prétraitement. Mais, dans le cas de gros calculs, les données prétraitées sont transmises au gros ensemble pour obtenir les résultats définitifs.

Cette conception d'utilisation d'ordinateurs en temps réel pour l'acquisition et le prétraitement et en temps différé pour l'exploitation des données a guidé les responsables de l'Institut Max von Laue - Paul Langevin (signe : ILL) dans le choix du système d'automatisa-

tion des expériences de diffraction neutronique. En effet, deux ordinateurs T 2000 Télémécanique assurent la gestion en temps réel de 12 expériences pour l'acquisition et le prétraitement des données. Les résultats sont stockés sur bandes magnétiques et peuvent être ainsi exploités dans des centres de calcul scientifique.

Le gain de temps d'une telle organisation est considérable. En effet, les données brutes d'une expérience ne sont pas très explicites et il est impossible au meilleur physicien de savoir si son expérience s'est déroulée comme il le voulait. A l'ILL, les ordinateurs T 2000, par leur prétraitement et grâce à un dispositif de visualisation Tektronix T 4002, donnent aux physiciens une première image de leur expérience et leur permettent de décider de la validité ou de l'intérêt de ce qu'ils viennent de faire. Ainsi seuls les résultats significatifs sont transmis et traités dans les centres de calcul, d'où gain de temps assez conséquent.

C'est le Laboratoire d'Electronique et de Technologie de l'Informatique (L.E.T.I), au sein du C.E.N.G., qui a été chargé de réaliser un tel système pour l'ILL.

Le but de la présente étude est donc de décrire le système informatique, baptisé CARINE (Computer Aided Research in Neutron Experiments), qui gère les expériences de diffraction neutronique, système original par :

- l'utilisation du CAMAC comme système de transmissions de données,
- l'utilisation d'un FORTRAN temps réel,
- une très grande sécurité de fonctionnement,
- un temps de réponse très réduit.

Bien entendu, il n'était pas question de confier à une seule et même personne le soin de développer tout l'ensemble du système. Le système CARINE a été réalisé par une équipe d'ingénieurs et de programmeurs, aussi bien du LETI que de l'ILL, judicieusement conseillé par des ingénieurs de la Société Télémécanique. L'auteur du présent rapport a participé à ce travail d'équipe et a réalisé la partie du système global qu'on lui avait confiée. Cependant, il a voulu décrire l'intégralité du système, en portant, bien sûr, l'accent sur ses propres travaux, afin de faciliter la compréhension des lecteurs et en sorte que le présent document puisse constituer un manuel de référence pour le système CARINE ainsi développé.

Lorsque le projet CARINE a débuté, l'équipe chargée de le réaliser avait en main les données suivantes :

- on disposait d'un certain matériel, fourni par la Société Télémécanique, qu'il fallait utiliser au mieux de ses possibilités.
- associé à ce matériel, nous disposions d'un software de base, ainsi que d'une version d'essai du superviseur temps réel RTDMS de la Société Télémécanique, gracieusement prêtée par la dite Société, et d'un FORTRAN autonome.

Comme nous le verrons par la suite, il est apparu que la réalisation du système allait comprendre trois grosses parties :

- Adaptation du FORTRAN aux problèmes temps réel
- Ecriture d'un interface entre le système Télémécanique et le système CAMAC de transmission de données,
- Réalisation d'un système de commande pouvant gérer simultanément plusieurs utilisateurs et contrôler le déroulement de leurs programmes.

Ces trois idées directrices ont donc permis de découper le travail à faire et de le confier à trois sous-équipes, sachant bien sûr qu'il y aurait un premier travail de définition des interfaces communs aux divers programmes à effectuer ensemble dans un premier temps. Donc, après définition des données communes permettant à chacun d'avoir les bases nécessaires pour travailler dans sa direction, la répartition des tâches s'est faite comme suit :

1) Monsieur VINIT, Ingénieur au LETI, avec Monsieur LESOURNE, docteur-ingénieur à l'ILL, devaient se mettre en rapport avec Messieurs OUDJAOUDI et LUMINEAU, de la Société Télémécanique, afin de prévoir et réaliser les modifications nécessaires pour placer FORTRAN sous le contrôle de RTDMS dans l'optique d'une utilisation en Temps Partagé. Monsieur VINIT était plus particulièrement chargé des problèmes de Temps Partagé, ainsi que des Entrées/Sorties FORTRAN, alors que Monsieur LESOURNE devait résoudre le problème des bibliothèques de programmes sur disque.

2) Monsieur DARIER, ingénieur au LETI, devait se charger de la réalisation d'un moniteur d'E/s pour le système CAMAC, ainsi que de l'Interface avec le FORTRAN.

3) L'auteur du rapport devait s'occuper des autres problèmes, c'est-à-dire essentiellement :

- gestion simultanée de plusieurs télétypes ainsi que des autres périphériques, donc d'étudier dans le détail les E/s sous RTDMS
- création d'un langage de commande permettant à l'utilisateur de contrôler le déroulement de ses programmes FORTRAN
- réalisation d'un système de VISUALISATION conversationnel permettant au physicien de travailler aisément sur les courbes représentatives de ses résultats.

Donc, dans les chapitres qui vont suivre, bien que les descriptions des diverses parties du système soient de la main de l'auteur, il ne faut pas oublier que leur réalisation effective et leur conception peuvent être le fait des autres équipes du système, l'auteur du rapport ayant réalisé le système COSYNUS, groupant le langage de commande et la gestion du DISPLAY.

A - DESCRIPTION D'UNE EXPERIENCE TYPE DE DIFFRACTION NEUTRONIQUE

Il est hors de propos d'entamer ici un cours de diffraction neutronique, mais il apparaît comme indispensable de préciser quels sont les buts de cette science, comment elle procède, ainsi que de définir un certain vocabulaire, afin de faciliter la compréhension de ce qui va suivre. Il ne faut pas oublier que le système CARINE a été conçu pour répondre aux besoins des physiciens et que ce n'est qu'après une longue enquête auprès d'eux que les propositions concrètes sur les réalisations, tant hardware que software, ont vu le jour.

I - PRINCIPES DE LA DIFFRACTION NEUTRONIQUE

Les physiciens utilisent les propriétés de la diffraction neutronique pour essayer d'approfondir leurs connaissances sur la structure de la matière ; cette technique permet en effet d'avoir des résultats très précis sur la constitution d'un cristal quant à la disposition relative des atomes et aux forces inter-atomiques qui assurent la cohésion de l'ensemble.

On appelle <u>maille</u> du cristal le plus petit motif, c'est-à-dire arrangement d'atomes, qui permet de représenter le cristal par simple juxtaposition. Une maille est connue lorsqu'on connaît les longueurs des vecteurs unitaires de la maille ainsi que leurs angles respectifs.

- a, b, c, pour les longueurs
  α, β, γ, pour les angles

Le cristal étant constitué d'une multitude de mailles juxtaposées, toutes les origines des mailles jouent le même rôle et portent le nom de <u>noeuds</u> du réseau. Un plan défini par trois noeuds en contient une infinité : c'est un <u>plan réticulaire</u>.

Trois lettres reviennent très souvent en diffraction neutronique : ce sont les <u>indices de Miller</u> h, k, l. Les indices de Miller constituent les coordonnées des noeuds dans un système ayant pour vecteurs unitaires les vecteurs de la maille a, b, c. Ainsi un plan réticulaire peut être défini par ses trois indices h, k, l.

La loi fondamentale de la diffraction neutronique est la <u>loi de Bragg</u>. Un faisceau de neutrons d'énergie donnée (donc de longueur d'onde donnée) arrivant sur un plan réticulaire (h, k, l) avec un angle d'incidence θ sera réfléchi par ce plan si l'on a :

$$\lambda = 2\, d_{hkl} \sin\theta \qquad \text{où } \lambda = \text{longueur d'onde neutron}$$

$$\frac{1}{d^2_{hkl}} = \frac{h^2}{a^2} + \frac{k^2}{b^2} + \frac{l^2}{c^2}$$

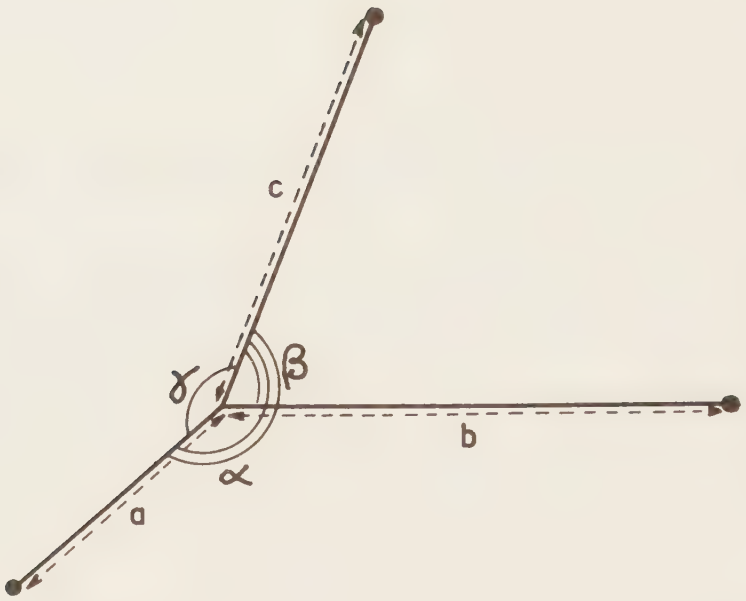


Figure N° 1

Schéma de la maille d'un cristal

Le principe de la diffraction neutronique est donc d'envoyer un faisceau de neutrons de même longueur d'onde sur un cristal et d'étudier la répartition en fonction de l'angle d'incidence θ des neutrons réfléchis. On doit obtenir une courbe comportant plusieurs pics correspondants aux réflexions par les divers plans h k l du réseau cristallin.

Connaissant λ, h, k, l, on peut donc en tirer les valeurs de a, b, c, donc approfondir la structure du cristal expérimenté.

Certains cristaux de structure parfaitement connue servent de référence.

II - MONTAGE PRATIQUE D'ETUDE D'UN CRISTAL

Le paragraphe précédent permet d'envisager quels sont les dispositifs nécessaires à la réalisation d'une expérience de diffraction neutronique. Il faut disposer :

- d'une source de neutrons : c'est le plus souvent un réacteur nucléaire ou "pile". Plus le flux des neutrons incidents est important, plus la durée d'une expérience est courte.
- d'un dispositif permettant de sélectionner une seule longueur d'onde dans le flux de neutrons incidents : c'est le monochromateur. Il s'agit d'un cristal dont la maille est parfaitement connue et qui sert à isoler par réfraction les neutrons de longueur d'onde λ_o telle que :

$$\lambda_o = 2\ d \sin \theta_o$$

- d'un dispositif permettant de pouvoir orienter le cristal échantillon à étudier dans toutes les directions de l'espace : c'est le goniomètre
- d'un dispositif permettant d'isoler parmi les neutrons diffractés par l'échantillon ceux de longueur d'onde λ_o : c'est l'analyseur dont le fonctionnement est identique à celui du monochromateur.

- d'un appareil permettant de compter les neutrons sortant de l'analyseur : il s'agit d'un compteur associé à une échelle de comptage.

Le schéma-type d'une expérience est donc celui donné à la figure N° 2.

On voit donc que le matériel nécessaire à une expérience est assez complexe et se compose :

- de moteurs : permettant de positionner les divers angles du montage à une valeur bien précise (en général au 1/100 de degré)
- de codeurs : permettant de vérifier la valeur de l'angle de l'axe auquel ils sont couplés
- d'échelles de comptage associées à des détecteurs de particules et effectuant la statistique des neutrons diffractés
- de dispositifs de chauffage ou de cryogénie permettant de faire varier la température de l'échantillon, température qui peut être lue à tout moment
- de dispositifs de "flipping", c'est-à-dire de variation alternée du sens de rotation du neutron sur lui-même, grâce à un champ magnétique alternatif appliqué au faisceau de neutrons incidents.

III - AUTOMATISATION D'UNE EXPERIENCE DE DIFFRACTION NEUTRONIQUE

Dans la plupart des cas, le déroulement type d'une expérience de diffraction neutronique est le suivant : après avoir positionné les divers paramètres de l'expérience en fonction de la raie (h, k, l) à étudier, le physicien fait varier un des angles par pas constant de façon à balayer une certaine zone angulaire autour de la raie considérée, et pour chaque point, effectue le comptage des neutrons diffractés par l'échantillon.

Les comptages à chaque point peuvent se faire selon deux critères :

- Précompte moniteur : l'expérience possède un détecteur qui compte le nombre de neutrons arrivant sur l'échantillon. Ce détecteur s'appelle le moniteur, et dans le cas d'un précompte moniteur, le comptage en un point s'arrête lorsque le nombre de neutrons détectés par le moniteur atteint une valeur pré-fixée.
- Prétemps : l'expérience possède un dispositif permettant de fixer la durée du comptage en un point à une valeur choisie par le physicien.

Pour automatiser entièrement une telle expérience, il faut :

- pouvoir programmer la partie mécanique du dispositif afin de pouvoir effectuer des balayages (appelés aussi SCANS) sur les angles,
- pouvoir gérer l'acquisition des taux de comptage en chaque point, et ce en fonction du critère choisi pour l'expérience : précompte moniteur ou prétemps.
- pouvoir faire des calculs assez complexes afin de calculer les paramètres d'un SCAN en fonction des données d'entrée (maille du cristal, raie h, k, l, etc....)
- pouvoir visualiser assez rapidement les résultats d'une expérience, afin de pouvoir déterminer s'ils sont ou non significatifs.

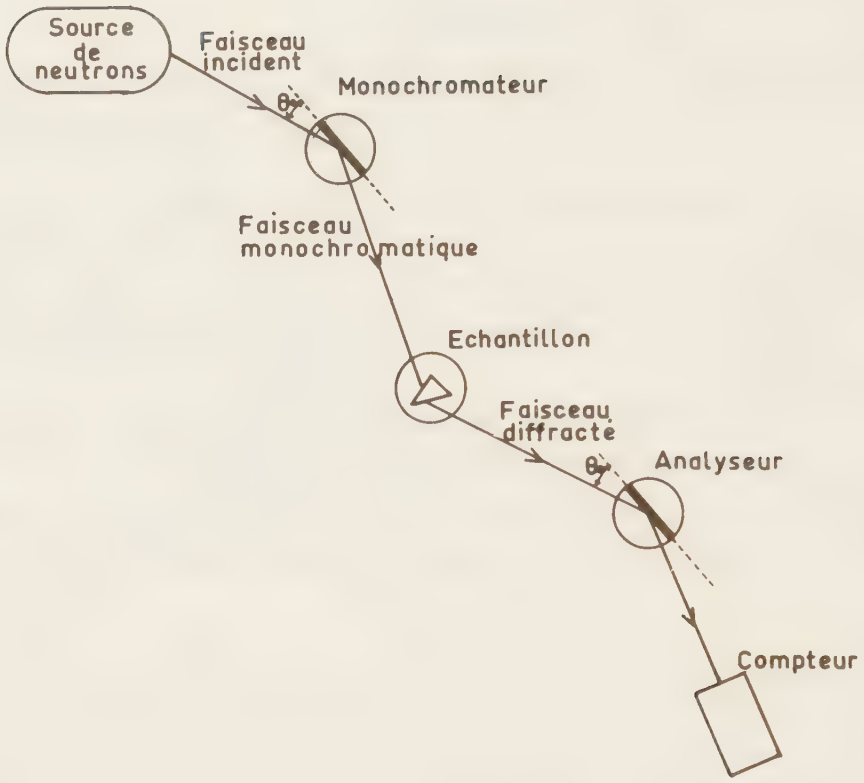


Figure N° 2

Schéma-type d'une expérience de diffraction neutronique

La première étape dans l'automatisation de telles expériences s'est réalisée à l'aide de circuits logiques câblés, réalisant la séquence des opérations nécessaires au déroulement de l'expérience. Des analyseurs multicanaux donnaient immédiatement l'allure de la courbe. Mais, comme tout ce qui est logique câblée, il n'était pas question de vouloir changer le déroulement, sinon tout était à reprendre.

L'apparition sur le marché de petits ordinateurs bon marché et assez rapides a changé les données du problème. La logique câblée a fait place à la logique programmée avec tous les avantages qu'elle comporte quant à la souplesse et aux facilités de calcul. C'est ainsi que, dans un premier temps, on a commencé par faire gérer par l'ordinateur la phase d'acquisition et de traitement des données, pour déboucher par la suite sur la gestion complète de l'expérience en laissant l'ordinateur contrôler également tous les mouvements mécaniques de l'appareillage.

Bien entendu, un tel progrès ne s'est pas fait sans certains désagréments : en particulier, le physicien est devenu informaticien et a dû se familiariser avec les techniques de programmation, aussi bien au niveau assembleur qu'au niveau langage évolué. Afin de faciliter sa tâche, les dernières années ont vu se développer une multitude de systèmes axés sur ordinateurs de taille moyenne permettant la programmation en langage simple de l'expérience, accessible à tout utilisateur ayant des notions de langages évolués tels que le FORTRAN.

IV - QUALITES REQUISES POUR UN LANGAGE ORIENTE VERS LA DIFFRACTION NEUTRONIQUE

Si l'on prend en compte toutes les considérations qui précèdent, on s'aperçoit qu'un langage orienté vers la diffraction neutronique doit répondre aux impératifs suivants :

- ce doit être un langage de calcul : il ne faut pas oublier qu'en diffraction neutronique on a besoin de calculs très précis, non pas seulement dans la phase de traitement des résultats, mais aussi au cours du déroulement de l'expérience, pour positionner les angles de l'appareillage en fonction des données d'entrée.
- Il doit pouvoir permettre la gestion facile de l'appareillage : en effet, le physicien doit pouvoir gérer le déroulement complet de son expérience, c'est-à-dire essentiellement :
  . positionner des moteurs,
  . lancer et arrêter un comptage en fonction d'un critère (prétemps ou précompte),
  . lire, positionner ou remettre à zéro des échelles de comptage.

  Ce langage doit donc permettre un dialogue facile avec l'appareillage.
- Il doit être souple et sûr

  la souplesse peut être assurée par une conception modulaire du système software : le physicien a à sa disposition un certain nombre de modules effectuant des opérations complexes mais répétitives telles que :
  . positionnement d'un ou plusieurs moteurs,
  . SCAN sur un ou plusieurs angles,
  . acquisition automatique,
  . corrections de courbe.

  Le physicien "joue" avec ces divers modules pour définir son expérience, les modules jouant le rôle de pièces d'un "Meccano" software.

  La sécurité de fonctionnement est indispensable car une anomalie pourrait entraîner la destruction de l'appareillage ou la perte de résultats accumulés sur plusieurs jours. Il faut donc :
- guider le physicien sans l'enfermer dans un carcan : un système conversationnel convient parfaitement
- vérifier systématiquement toutes les valeurs entrées par le physicien ou provenant de calculs
- arrêter le déroulement du programme dès qu'une alarme quelconque est déclenchée, sans rien perdre des résultats déjà acquis.

Tout ce qui précède permet déjà d'entrevoir quelles pourraient être les solutions envisageables.

. un langage évolué du style FORTRAN conviendrait parfaitement quant à ce qui concerne la partie calcul. Malheureusement, un tel langage est absolument impraticable quant à la gestion temps réel de l'appareillage.

. un langage niveau Assembleur pourrait convenir quant aux problèmes de temps réel et de gestion d'appareillage. Il pourrait également, en lui adjoignant des facilités comme l'Arithmétique Flottante, faire des calculs complexes. Mais alors, comme on travaille au niveau

de l'instruction machine, le physicien doit devenir un informaticien de valeur, et il n'est plus question de parler de souplesse pour un tel système.

. une solution mixte, c'est-à-dire langage évolué pour les calculs, auquel on incorpore la gestion temps réel de l'appareillage, semble être la solution idéale, mais il faut être prudent et faire la part des choses ; en particulier, il ne faut pas oublier que l'on dispose d'un certain hardware, d'un certain contexte d'utilisation et qu'il faut peser le pour et le contre de chaque solution avant de se lancer à fond dans la définition d'un tel système.

V - SOLUTION ADOPTEE DANS LE CADRE DU SYSTEME CARINE

Comme nous venons de le dire, il ne faut pas négliger le contexte "hardware" ou utilisation d'une telle réalisation. Une description détaillée du "Hardware" utilisé sera donnée ultérieurement. Pour l'instant rappelons simplement que :

- l'appareillage de l'expérience est connecté à l'ordinateur (un T 2000 Télémécanique) par un système de transmission de données nouveau le CAMAC. Un coupleur spécial T 2000-CAMAC a été développé et il faut se plier à ses contraintes.
- l'ordinateur doit être connecté à 6 expériences pouvant fonctionner simultanément et il ne faut pas oublier le côté "multiprogrammation" du système, aussi bien pour les Entrées/Sorties classiques, que pour celles du système CAMAC.

Une première solution retenue était celle de la création d'un langage particulier, interprété lors de l'exécution et enchaînant une suite de modules, correspondant à des instructions spécialisées, permettant de gérer le déroulement de l'expérience et d'assurer les séquences de calcul classiques en diffraction neutronique. C'est là que résidait le défaut d'une telle proposition : on n'autorisait pas tous les calculs du FORTRAN, mais seulement certains calculs complexes, bien figés, et ce, pour ne pas avoir à écrire un nouveau FORTRAN sur le T 2000. Cette solution était trop rigide quant à la partie calcul et a du être abandonnée.

Dans la "panoplie" Software de l'ordinateur envisagé figurait :

- un FORTRAN autonome, de type interprétatif, et assez puissant, avec même certains fonctions développées spécialement pour permettre d'accéder jusqu'au bit de l'instruction ou de la donnée.
- un superviseur temps réel R.T.D.M.S. (Real Time Disk Monitor System) permettant une Multiprogrammation facile, mais au niveau programmes assembleurs
- un Moniteur d'E/S MONES gérant les E/S sur tous les périphériques de la gamme classique de l'ordinateur.

Considérant toutes ces données, la solution définitivement adoptée est la suivante :

- Le programme gérant le déroulement de l'expérience est un programme FORTRAN. Afin de pouvoir commander aisément l'appareillage, une bibliothèque spéciale de fonctions standards, directement accessibles à partir du FORTRAN, sera développée.

L'ensemble de l'exécutif du FORTRAN sera contrôlé par R.T.D.M.S. afin d'assurer une gestion temps réel de l'expérience, avec un système de time-sharing pour

autoriser la multiprogrammation. Pour ce faire :

. il faut créer l'interface R.T.D.M.S. - FORTRAN, ainsi que l'interface FORTRAN-CAMAC pour gérer en temps réel toutes les Entrées-Sorties, classiques ou non

. il faut créer un moniteur d'Entrées-Sorties CAMAC compatible avec RTDMS et fonctionnant similairement à MONES

. il faut créer un système de gestion en temps partagé du FORTRAN, sous le contrôle direct de RTDMS

. il faut développer un système de commande, directement contrôlé par RTDMS et pouvant se dérouler de façon indépendante du FORTRAN, destiné à :

- contrôler le déroulement des divers programmes FORTRAN
- effectuer les diverses fonctions système indispensables
- gérer l'unité de DISPLAY connectée à l'ordinateur.

Le système global peut donc se schématiser selon le modèle de la figure n° 3.

Cette solution, bien que moins séduisante pour l'esprit que celle du langage orienté, présente les avantages suivants :

- utilisation de toutes les possibilités de calcul du FORTRAN
- utilisation d'un FORTRAN déjà existant et donc seulement à adapter aux problèmes temps réel

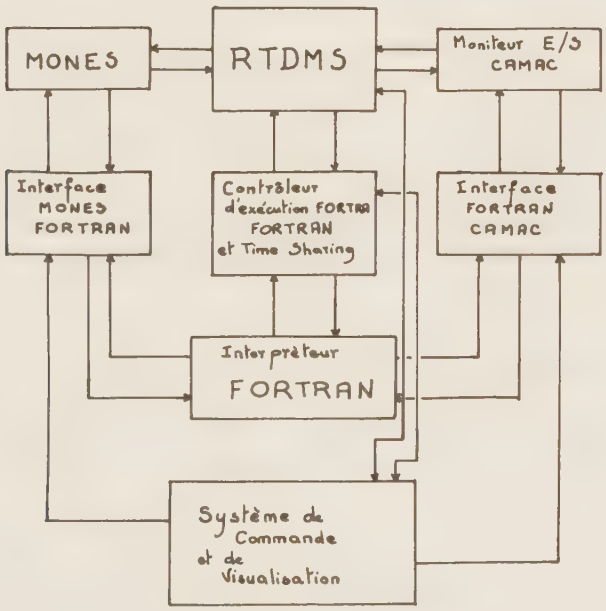


Figure N° 3

Schéma général du système informatique " CARINE "

- commande de l'appareillage de l'expérience souple et aisée grâce à des fonctions standards rajoutées au FORTRAN
- contrôle total et facile du déroulement des programmes grâce à un système de commande
- temps de réponse très bref et sécurité de fonctionnement grâce à un contrôle direct de RTDMS sur tous les programmes en cours.

B - DESCRIPTION DU HARDWARE CONNECTE AU SYSTEME CARINE

Lorsqu'on veut créer un système informatique pour contrôler un certain processus, plusieurs critères rentrent en ligne de compte, et c'est celui ou ceux considérés comme plus importants qui imposent l'idée directrice du projet.

Ceux qui reviennent le plus souvent sont :

- la nature du processus à contrôler : selon les types de processus les modules software seront différents et surtout c'est l'interface entre l'ordinateur et le processus, qui n'est pas un périphérique standard, qui va beaucoup changer.
- l'utilisation : l'ordinateur est mis à contribution pour faciliter la tâche de l'utilisateur du processus. Il faut donc une enquête approfondie auprès de celui-ci afin de répondre à ses désirs, et surtout que ce ne soit pas l'utilisateur qui soit exclave de l'ordinateur.
- le Hardware de contrôle : il faut choisir l'ordinateur le mieux adapté au processus, soit par l'étendue de sa bibliothèque de programmes, soit par sa large gamme de périphériques standards. Cependant, il est à noter que le matériel peut être plus ou moins imposé par des motifs d'ordre économique (fourchette de prix, réutilisation de matériel déjà existant, etc...)

C'est pourquoi, même si lors de la définition de la philosophie générale d'un système, on analyse en faisant abstraction du matériel utilisé, il est indispensable, à l'étape de la conception pratique, de tenir le plus grand compte du matériel qui va servir de support au système, afin d'assurer une efficacité maximale. Pour cette raison, la description du matériel utilisé dans le système CARINE s'avère indispensable si l'on veut comprendre la structure de l'ensemble software.

I - LE SYSTEME DE TRANSMISSION DE DONNEES "CAMAC"

Dès que l'ordinateur a été utilisé pour piloter un appareillage quelconque, un problème important a vu le jour : celui du couplage.

Dans un premier temps, on s'est attaché à réaliser une interface permettant l'adaptation aussi bien de la logique que des niveaux utilisés. Mais de nombreux inconvénients sont alors apparus, car si l'on voulait modifier quelque chose côté ordinateur ou côté appareillage, ou si l'on voulait transplanter simplement l'appareillage, de nombreux problèmes intervenaient qui faisaient renoncer à de tels projets. Ce type de difficultés s'est surtout présenté dans l'instrumentation nucléaire où de nombreuses expériences se déplaçaient d'un centre d'études à l'autre.

Les électroniciens des divers pays d'Europe concernés se sont alors réunis

pour définir un système de transmission de données standardisé tant du côté logique que du côté mécanique et destiné à être couplé à un ordinateur. Tous les appareillages déjà existants seraient repensés sous forme de "Modules CAMAC" absolument interchangeables. Ainsi, l'interface entre l'ordinateur et l'appareillage serait considérablement simplifiée. En cas de changement d'ordinateur, seul un nouveau coupleur ORDINATEUR-CAMAC serait à repenser, et vu sa simplicité, le serait rapidement et à peu de frais.

I.1 - Principe général de fonctionnement

Le système CAMAC a été conçu principalement dans l'optique d'une instrumentation modulaire orientée vers l'automatisation du traitement des données.

Il a pour but :

- de faciliter les échanges de données numériques entre l'ordinateur et des tiroirs fonctionnels,
- de permettre l'utilisation généralisée des circuits intégrés,
- de permettre l'utilisation de "cartes" ou de "tiroirs" accomplissant une fonction bien précise.

Ainsi donc les éléments fonctionnels sont réalisés sous forme de tiroirs. Ces tiroirs sont enfichés dansles emplacements d'un chassis, qui assure la liaison avec le reste du système. Chaque emplacement du chassis est défini par une adresse. Un tiroir peut occuper plusieurs emplacements ; dans ce cas, il n'a qu'une seule adresse, qui est celle de l'emplacement le plus à gauche vu de face. Chaque chassis comporte 24 emplacements, numérotés de 1 à 24. A l'extrême droite du chassis, un 25è emplacement est prévu pour recevoir un module spécial appelé "contrôleur de chassis", qui assure le dispatching des signaux intéressant le chassis.

Plusieurs chassis peuvent être connectés en parallèle pour constituer une branche. Dans ce cas, chaque chassis est défini par son adresse sur la branche. Chaque branche est gérée par un "contrôleur de branche". Ces derniers sont regroupés dans un chassis, appelé chassis système, qui comporte le coupleur avec l'ordinateur.

La structure est donc celle représentée sur la figure N° 4.

I.2 - Lignes utilisées dans l'envoi d'un ordre CAMAC

Un certain nombre de lignes transitent à travers les branches et les chassis d'un système CAMAC.

On y trouve :

- 5 lignes d'adresse station N_1 à N_5 qui assurent l'adressage à l'intérieur du chassis. Il est à noter que ces 5 bits sont décodés au niveau du contrôleur chassis, lequel est relié par lignes individuelles aux différentes positions. Les combinaisons permises par les 5 bits d'adresse sont supérieures en nombre aux emplacements : les adresses au-delà de 24 sont utilisées pour réaliser des fonctions bien précises dans le contrôleur de chassis.
- 4 lignes de sous-adresse : A_1, A_2, A_4, A_8, qui assurent l'adressage à l'intérieur d'un module fonctionnel. En effet, certains modules un peu complexes peuvent nécessiter l'utilisation de

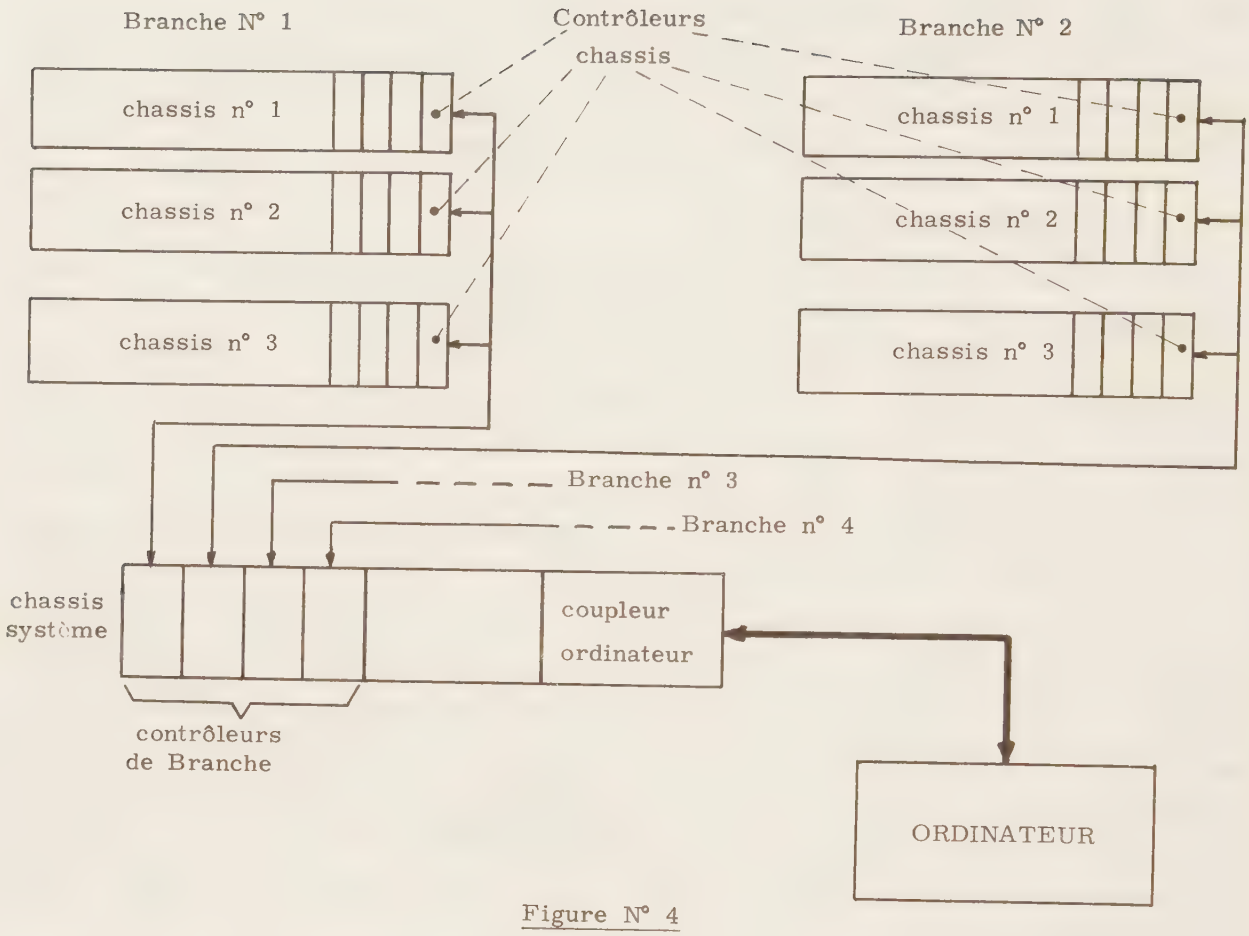


Figure N° 4

Structure type d'une installation CAMAC

sous-adresses afin de préciser à quelle partie on s'adresse (cas de plusieurs registres par exemple)

- 5 lignes de fonctions : F_1 F_2 F_4 F_8 F_{16} qui permettent de spécifier jusqu'à 32 types de fonctionnement pour un module.
- 48 lignes de données proprement dites, qui se répartissent en :
  . 24 lignes omnibus de lecture, c'est-à-dire, transmission vers l'extérieur (R_1 à R_{24})
  . 24 lignes omnibus d'écriture, c'est-à-dire envoi de données en provenance de l'extérieur (W_1 à W_{24})

Tous les signaux qui précèdent sont adressés, c'est-à-dire destinés à une seule station sélectionnée par N et A. Certains signaux sont non-adressés, c'est-à-dire qu'ils s'adressent à tous les modules présents. Ce sont :

- Z : signal d'initialisation
- I : signal d'inhibition
- C : signal de remise à zéro

- B : signal d'occupation
- Q : signal de réponse

De plus, chaque module d'un chassis est relié par une ligne spéciale d'appel au contrôleur de chassis. Dans une branche, les appels des contrôleurs de chassis sont regroupés, et génèrent au niveau du chassis système un appel qui est destiné à signaler à l'organe externe (ici, l'ordinateur) qu'un module est prêt à émettre ou à recevoir des données.

Enfin, dans le cas d'un système complet, on a besoin également de :

- 3 bits adresse chassis dans la branche : C_1 C_2 C_4
- 3 bits adresse branche : BR_1 BR_2 BR_4

Ainsi on peut connecter jusqu'à 8 chassis par branche sur 8 branches (en fait, il n'y a que 7 chassis par branche car l'adresse chassis "0" est réservée pour désigner le chassis système).

I.3 - Fonctions courantes du système CAMAC

Nous avons vu que l'on pouvait définir jusqu'à 32 fonctions CAMAC différentes. Dans un souci d'uniformisation, le comité CAMAC a fixé le code de certaines fonctions qui reviennent souvent, les codes libres restant à la disposition de l'utilisateur.

On a ainsi les fonctions :

- lecture qui sont caractérisées par $F_{16} = 0$ $F_8 = 0$
- écriture qui sont caractérisées par $F_{16} = 1$ $F_8 = 0$
- exécution qui sont caractérisées par $F_8 = 1$

. contrôler un appel	F = 8
. remettre à zéro un registre	F = 9 ou F = 11
. remettre à zéro un appel	F = 10
. mettre hors service	F = 24
. incrémenter un registre	F = 25
. mettre en service	F = 26
. contrôler un état	F = 27

I.4 - Ordre CAMAC

L'ordre CAMAC peut se comparer à une instruction dans un ordinateur.

Lorsque l'on veut activer un module fonctionnel pour lui faire accomplir une certaine fonction

1) on sélectionne le module en donnant :
   - une adresse branche Br
   - une adresse chassis C
   - une adresse station N
   - une sous-adresse A

2) on lui envoie le code fonction F correspondant à l'action choisie.

S'il y a des données à lire ou à écrire, elles sont présentes sur les lignes R ou W.
Si le code fonction est un test, le résultat du test est transmis sur la ligne Q.
La transmission d'un ordre CAMAC complet nécessite donc l'emploi d'au moins 45 lignes.

I.5 - Description sommaire du coupleur CAMAC - T 2000

Lorsqu'on veut connecter le système CAMAC à un ordinateur, un problème se pose au niveau de la distribution des 45 signaux nécessaires à l'envoi d'un ordre CAMAC. Un coupleur est donc nécessaire, qui jouera principalement un rôle de tampon, mémorisant plusieurs informations en provenance de l'ordinateur et les envoyant toutes à la fois sur les diverses lignes CAMAC.

On a donc :

- un registre tampon qui mémorise les 24 bits d'information CAMAC en entrée ou en sortie
- un registre tampon qui mémorise les bits nécessaires à l'envoi d'un ordre CAMAC, c'est-à-dire :
  - . A 4 bits
  - . N 5 "
  - . Br 3 "
  - . C 3 "
  - . F 5 "
- en outre la ligne Q de réponse CAMAC est mémorisée.

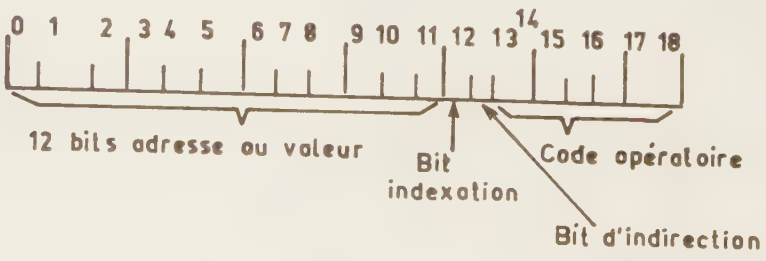
II - MATERIEL T 2000 UTILISE SUR CARINE

Lorsqu'on réalise un système informatique, on essaie de s'affranchir un peu du matériel utilisé afin de dominer le problème et avoir une plus large ouverture d'esprit. Mais ce matériel existe et c'est avec lui que l'on travaille : il ne faut donc pas l'ignorer, et c'est pourquoi une description du matériel T 2000 s'impose, d'autant plus que dans un système temps réel le hardware a un rôle important.

Il n'est point question de donner ici une description détaillée de tous les appareils du standard T 2000, mais d'avoir un aperçu général de l'ensemble, et de voir le système d'E/S afin que les pages suivantes soient compréhensibles.

II.1 - Unité centrale

L'ordinateur T 2000 possède une mémoire de capacité extensible de 4 K à 32 K, de cycle de base 1,5 µs. La longueur du mot est de 19 bits. Il possède 12 registres dont 8 accessibles par programme. Une instruction machine est de longueur fixe et tient sur un mot. Le format est le suivant :



La mémoire est divisée en 8 blocs de 4 K

Les modes d'adressage sont les suivants :

- adressage immédiat : la partie adresse doit être considérée comme une valeur
- adressage direct : la partie adresse représente une adresse dans un bloc de 4 K
- adressage indirect
- adressage indexé : le registre index est désigné par X

Les principaux registres utilisés sont :

- A : accumulateur 19 bits
- B : extension du registre accumulateur
- P : pointeur (compteur ordinal)
- X : registre index
- E : registre d'état (Masque des interruptions internes et externes)
- T : registre tampon (sauvegarde de P si rupture)

L'ordinateur T 2000 possède tout un répertoire d'instructions, trop long à énumérer, mais analogue à celui des autres ordinateurs de cette catégorie.

Les instructions et les opérandes peuvent être dans des blocs distincts : on parle alors

- de bloc instruction (abrév. BI)
- de bloc opérande (abrév. BO)

La détection de fin de bloc instruction (c'est-à-dire le passage par zéro du registre P) provoque une alarme interne.

La mémoire possède une zone protégée délimitée par une barre de verrouillage accessible par programme. En dehors de cette zone, toute tentative d'exécuter une instruction privilégiée, c'est-à-dire :

- Entrée/Sortie
- Branchement vers zone protégée
- Rangement en zone protégée

provoque une alarme, qui peut être traitée.

II.2 - Fonctionnement des E/S

Les entrées-sorties avec les périphériques standards du calculateur T 2000 se rangent en trois catégories :

- Entrées-sorties programmées : les échanges se font entre le registre A et le périphérique et sont contrôlés par l'unité centrale.
- Entrées-sorties mode canal : un dispositif électronique, le canal, se charge des échanges entre le périphérique et la mémoire. L'unité centrale peut exécuter des instructions pendant ce temps-là, mais le dispositif d'accès à la mémoire est commun à l'unité centrale et au canal, donc le débit d'information n'est pas optimal.
- Entrées-sorties mode canal avec accès direct : le principe est le même que le précédent, mais il y a un dispositif multiplexeur qui permet l'accès simultané à la mémoire à partir de l'unité centrale et du canal. Le débit d'information s'en trouve donc augmenté.

Tout périphérique connecté à l'ordinateur T 2000 possède un coupleur spécifique placé dans un bac d'entrée-sortie. La position dans le bac détermine l'adresse du périphérique. Dans un bac, on peut connecter jusqu'à 15 périphériques. Chaque bac d'entrées-sorties possède également une adresse. L'adresse du bac d'E/S et l'adresse du périphérique dans le bac constituent l'adresse complète du périphérique.

De plus, on a la possibilité d'avoir 4 sous-adresses à l'intérieur d'un coupleur périphérique pour préciser la nature de l'ordre d'E/S :

- émission d'un caractère
- émission d'une fonction (mise Hors Tension, Mise sous tension, etc....)
- entrée d'un caractère
- entrée du mot état du périphérique

En fonctionnement en mode programmé, la validité de l'instruction d'E/S, c'est-à-dire la disponibilité du périphérique, est caractérisée par le fait que l'instruction qui suit est sautée. On peut également faire générer des appels prioritaires pour signaler que le périphérique est disponible : dans ce cas, l'instruction d'E/S doit toujours être valide.

Dans le cas du fonctionnement en mode canal, le rythme des échanges est directement géré par le canal. Ce dernier peut envoyer un appel prioritaire lorsque l'échange est terminé, pour alerter l'unité centrale.

II.3 - Fonctionnement des appels prioritaires

Nous allons décrire un peu plus en détail le fonctionnement des appels prioritaires générés par des périphériques fonctionnant en mode programmé. C'est le cas des organes d'E/S classiques :

- Télétypes
- Lecteur de ruban
- Lecteur de cartes
- Perforateur de ruban

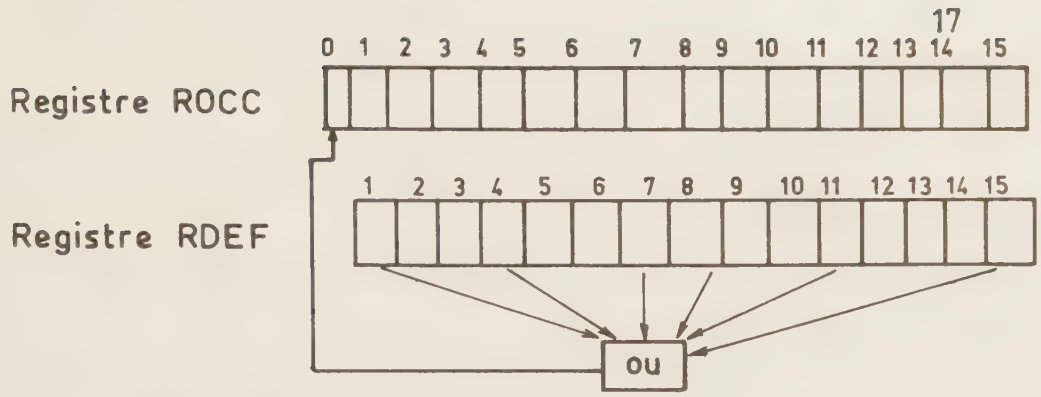
L'étude de ce type de fonctionnement est très importante car, dans le cas d'un fonctionnement en multiprogrammation, la fiabilité du système dépend beaucoup d'une bonne gestion des appels prioritaires.

Tout périphérique connecté à l'ordinateur peut générer deux sortes d'appels prioritaires :

- appel libération : indiquant que le périphérique est prêt à recevoir ou a une donnée à émettre.
- appel défaut : indiquant un mauvais fonctionnement du périphérique.

Les appels de chaque périphérique sont regroupés au niveau du bac d'E/S qui possède deux registres :

- le registre RDEF de 15 bits, qui mémorise les appels défaut des périphériques connectés
- le registre ROCC de 16 bits qui mémorise les appels libération des périphériques. Le bit 0 du registre ROCC est la réunion de tous les bits du registre RDEF, indiquant qu'il y a au moins un périphérique en défaut.



La position du bit dans RDEF ou dans ROCC correspond à l'emplacement du périphérique concerné dans le bac d'E/S.

Donc, lorsqu'on a un appel en provenance d'un bac d'E/S, la séquence classique de traitement est la suivante :

- acquisition du registre ROCC
- examen du bit 0 de ROCC
  - . si positionné, lecture de RDEF
  - . traitement du défaut
- identification du périphérique appelant et traitement approprié.

Dans le cas du fonctionnement en mode canal, la gestion est différente et un peu plus complexe, mais le principe de base reste le même, mis à part le fait qu'en mode canal, les appels sont générés pour un bloc de données, alors qu'en mode programmé, les appels sont générés à chaque caractère.

II. 4 - Fonctionnement des interruptions externes

Le programme exécuté par l'ordinateur peut être à tout moment interrompu par des signaux externes appelés interruptions. Ces interruptions peuvent être générées par le système d'Entrées/Sorties classiques de l'ordinateur, mais également par tout autre dispositif couplé convenablement au système T 2000.

Lorsqu'une interruption arrive (en vocabulaire T 2000 une UR), le programme en cours est arrêté, et le pointeur est sauvegardé dans le registre tampon T.

Le compteur d'instruction est forcé à une valeur adresse contenue dans une mémoire liée à l'interruption, adresse à laquelle on a le sous-programme de traitement de l'interruption. Lorsque ce dernier est terminé, le programme principal est repris là où il avait été interrompu.

Chaque bac d'Entrées-Sorties possède donc sa propre adresse d'UR, étant entendu que certains périphériques de ce bac peuvent avoir leur propre adresse d'UR. Ainsi, une UR à l'adresse N provoque la mise à jour du compteur ordinal à l'adresse contenue dans la mémoire d'adresse N.

Les mémoires utilisées pour les branchements sur UR sont implantées au début du bloc 0 de la mémoire.

Les UR ont une certaine priorité relative : les UR d'adresse la plus faible sont les plus prioritaires. Un programme de traitement d'UR ne peut être interrompu que par une UR de priorité supérieure. Les UR de priorité inférieure ou égale sont mémorisées et prises en compte lorsque le traitement de l'UR hiérarchiquement supérieure est terminé.

III - SYSTEME HARDWARE INSTALLE SUR CARINE

III. 1 - Description de l'installation

On trouvera sur la figure n° 5 un schéma de l'installation permettant d'avoir un point de vue global du système.

Le système CARINE a été prévu pour gérer 6 expériences de diffraction neutronique. Le choix du nombre d'expériences a été déterminé par des raisons de charge du système et de capacité mémoire et disque disponible.

L'ordinateur T 2000 utilisé possède une taille mémoire de 32 K de 19 bits. Il dispose de 3 bacs d'Entrées-Sorties :

- un programmé
- un canal
- un canal accès direct

A l'ordinateur sont connectés :

- un disque de 256 K mots de 19 bits
- trois dérouleurs TMZ de bandes magnétiques
- un lecteur de ruban
- un perforateur de ruban
- un lecteur de cartes
- 6 Télétypes utilisateurs
- un télétype système
- un télétype backgraund
- un chassis système CAMAC
- un display T 4002 tektronix

Les télétypes, le lecteur de ruban, le lecteur de cartes, le perforateur de ruban, le système CAMAC fonctionnent sur la liaison programmée. Le disque est en accès direct. Le display et les unités de bande fonctionnent en mode canal.

De plus, dans le bac E/S programmées, se trouve un cadenceur qui est un dispositif permettant de générer des UR à intervalles de temps fixes, la période des impulsions pouvant être accessible par programme.

III. 2 - Interruptions externes à traiter

Nous allons considérer maintenant quelles sont les diverses interruptions susceptibles d'être générées par le hardware du système CARINE. Toutes ces sources doivent être parfaitement connues, car ce sont elles qui vont activer l'ensemble software du système.

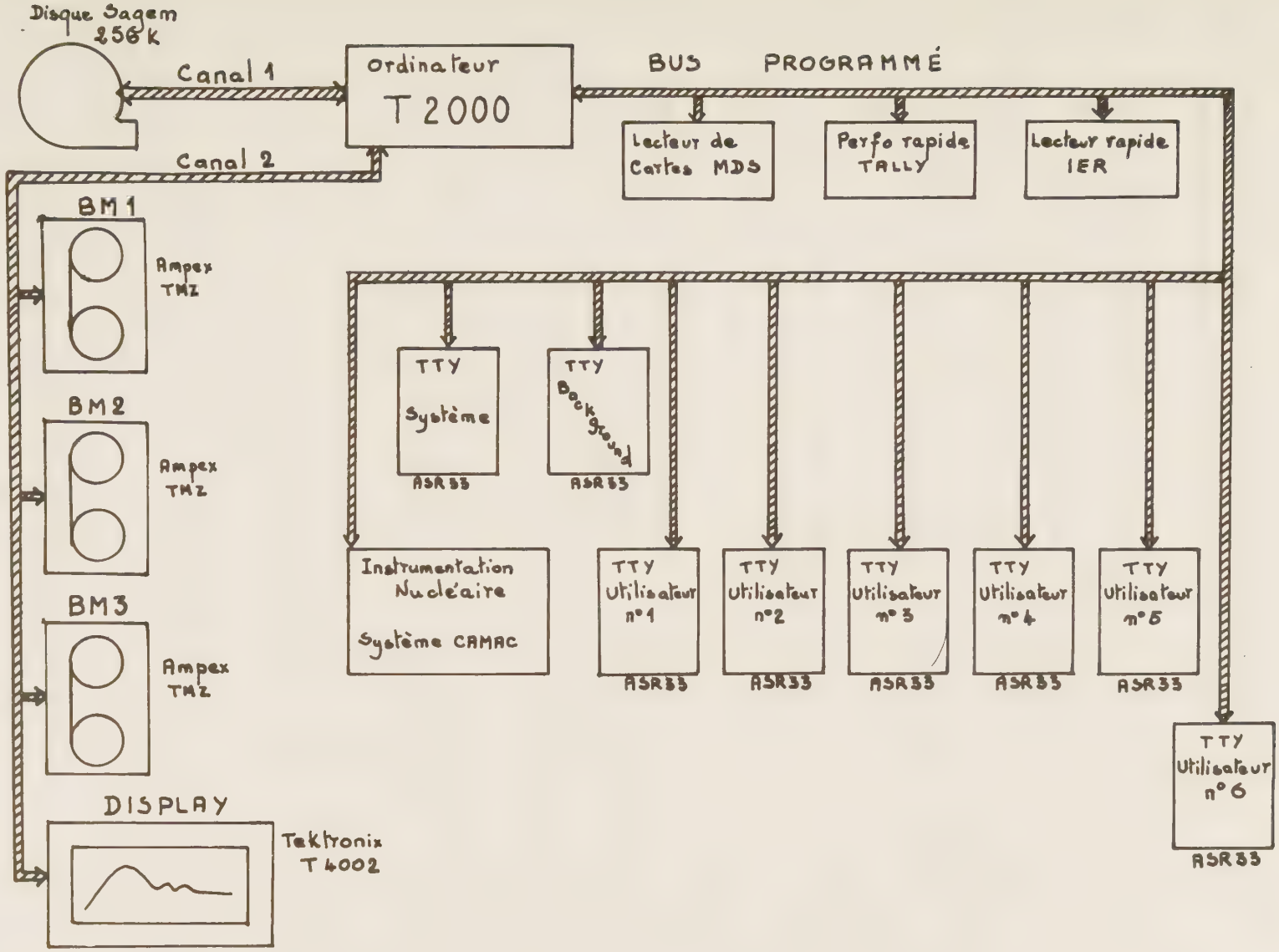


Figure N° 5

Organisation Hardware du Système CARINE

On a donc à traiter :

- l'UR du bac Entrées/Sorties programmées qui a lieu à l'adresse 15
- l'UR du bac canal disque, qui monte en 12
- l'UR du canal connecté aux bandes et au Display qui se produit en 13
- l'UR du cadenceur qui se produit en 11

En plus, le système CAMAC, bien que connecté aux Entrées/Sorties programmées génère 24 UR, réparties en 3 niveaux de priorité de 8 UR, d'adresses comprises entre 40 et 70.

Sur le premier niveau, on trouve les alarmes défaut secteur des diverses expériences.

Sur le second, on trouve les alarmes internes au système CAMAC.

Enfin, sur le troisième niveau sont connectés les appels des divers chassis CAMAC des expériences.

Bien entendu, à chaque niveau sont associés plusieurs périphériques et il faut les identifier à l'aide des registres ROCC et RDEF des bacs d'Entrées/Sorties.

III.3 - Masquage des interruptions

On a vu plus haut qu'un niveau d'interruption était interruptible par un autre niveau de priorité supérieure. Or, il est souvent nécessaire, dans le traitement d'un niveau d'interruption, de se protéger des interruptions pour éviter l'utilisation de tables de réentrance trop coûteuses en place mémoire.

Le calculateur T 2000 possède un registre d'état E qui permet de masquer les niveaux d'interruptions externes, ainsi que certains niveaux internes. Pour ce faire, il suffit de positionner à "1" le bit du registre E correspondant au niveau désiré. Dans ce cas, une interruption éventuelle sera prise en compte mais non traitée immédiatement : son traitement interviendra dès que le niveau correspondant aura été démasqué.

Bien entendu, il ne faut pas abuser d'un tel procédé, car le temps de réponse du système aux interruptions externes en souffrirait. Ce moyen est néanmoins très utile pour s'assurer qu'un appel externe ne peut pas venir perturber le déroulement de certaines séquences.

C - ENSEMBLE DES MODULES SOFTWARE CONSTITUANT LE SYSTEME CARINE

Nous nous sommes jusqu'à présent surtout attachés à la description des contextes qui ont influé sur la conception du système complet permettant de gérer le déroulement des expériences de diffraction neutronique de l'ILL.

Nous l'avons fait pour bien montrer quelles étaient les données matérielles que nous avions en main et quelles étaient les contraintes à prendre en compte.

Cependant, il ne faudrait pas conclure que le système a été développé en se limitant uniquement à répondre aux besoins du moment sans envisager soit des extensions ultérieures, soit des adaptations à d'autres problèmes. En fait, on peut très facilement élargir le problème à la gestion de processus quelconques avec un équipement quelconque. Nous allons essayer dans ce qui suit de montrer la généralité de ce qui a été fait, avant de rentrer dans la description effective des réalisations.

I - CONTROLE DE PROCESSUS

Préliminaires

Dans ce chapitre, nous allons essayer d'envisager comment on peut concevoir d'un point de vue plus général, la gestion par ordinateur d'un processus quelconque, conception générale guidant par la suite la réalisation effective du système. En particulier, cette vue d'ensemble est basée sur le fait, qu'à notre avis, le contrôle programmé d'un processus se passe, en général, à deux niveaux :

- au premier échelon, existe un système programmable, dont les programmes peuvent être écrits "sur mesure" en un langage de niveau quelconque, et dont les séquences d'instructions conditionnent la marche du processus selon le mode voulu.
- à un niveau plus élevé, nous trouvons un système de commande permettant l'activation du ou des programmes du niveau inférieur et permettant de modifier, en ligne, tous les paramètres du processus. C'est un système supervisant la marche des programmes contrôlant directement le processus.

Cette étude générale nous a semblé nécessaire en raison de l'apparition sur le marché d'ordinateurs de taille moyenne ou même de "mini-ordinateurs" qui ont changé les données dans beaucoup de domaines où l'emploi de systèmes informatiques apparaissait comme irréalisable à cause des prix de revient.

C'est ainsi que l'on trouve de plus en plus fréquemment l'utilisation d'ordinateurs dans le contrôle et l'exploitation de processus de tous ordres.

I.1 - Notions sur les processus

On appelle processus un ensemble organisé de diverses fonctions matérielles, effectuant un travail déterminé sous l'action d'ordres extérieurs, réagissant à certaines modifications internes et susceptibles d'envoyer vers l'extérieur des signaux que l'on peut ranger en trois catégories :

- réponses à des ordres reçus de l'extérieur
- indications sur l'état instantané du processus
- alarmes indiquant un mauvais fonctionnement

Un processus est donc connecté à un organe, que l'on peut appeler "pupitre de commande", d'où partent les ordres vers le processus et où aboutissent tous les signaux qu'il est susceptible de générer.

Le processus peut posséder une certaine "intelligence", en ce sens qu'il peut enchaîner les opérations nécessaires en réponse à une sollicitation du pupitre de commande, sans que ce dernier ait connaissance du détail du mécanisme, sinon par des alarmes éventuelles.

Nous retiendrons donc que, pour assurer la gestion du déroulement d'un processus, il faut :

- envoyer des ordres, c'est-à-dire des commandes
- pouvoir lire en permanence l'état du processus
- pouvoir visualiser et analyser les diverses alarmes possibles afin de provoquer les décisions adéquates.

La gestion d'un processus peut donc être assurée par un dispositif capable de prendre les décisions et d'envoyer les commandes nécessaires en réponse, soit à l'état du système, soit à des alarmes, pour maintenir le processus dans un état déterminé à l'avance. On appelle valeur de consigne la valeur que doit prendre un paramètre du processus, et qui doit rester constante en dépit des fluctuations des divers autres paramètres du système.

L'ordinateur semble parfaitement adapté à cette tâche, grâce à sa rapidité dans les calculs et dans les décisions.

I.2 - Contexte informatique pour un processus

Ayant donc pris la décision de faire gérer un processus par un ensemble informatique, nous allons essayer de voir ce qu'implique une telle décision.

Nous avons parlé plus haut d'un pupitre de commande où aboutissaient toutes les informations en provenance ou destinées au processus. L'ordinateur doit donc pouvoir accepter les signaux en provenance du processus et pouvoir générer des ordres acceptables par ce dernier. Or, le processus ne peut être considéré comme un périphérique classique et nécessite donc la présence d'une interface tant au niveau hardware qu'au niveau software, afin d'avoir une bonne gestion des E/S avec le dispositif à contrôler.

Le système informatique doit fonctionner en temps réel, car il est indispensable qu'il réagisse presque immédiatement aux changements d'état ou aux alarmes du processus. On doit donc posséder un superviseur temps réel capable de traiter les interruptions

générées par le dispositif et d'activer les programmes nécessaires avec éventuellement une certaine priorité.

Le système informatique doit permettre l'entrée des valeurs de consigne, leur modification éventuelle et la lecture aisée des divers paramètres du processus. De plus, certaines décisions relatives au fonctionnement du processus nécessitant l'enchaînement de plusieurs ordres élémentaires, il est intéressant de créer un dialogue entre le système de gestion proprement dit et l'opérateur, afin de simplifier au maximum la tâche de ce dernier.

Enfin, il est également intéressant d'introduire un certain symbolisme, en utilisant un langage proche du langage courant, afin que l'opérateur ne soit pas nécessairement un spécialiste. L'utilisation de tables système indiquant les diverses correspondances se révèle donc très pratique. Le système doit alors permettre la modification facile de telles tables lorsque le matériel évolue.

I.3 - Vue d'ensemble sur un système informatique de gestion de processus

Les diverses considérations du paragraphe précédent permettent de définir les grandes lignes d'un système gérant la marche d'un processus. Un tel système comporte :

- un superviseur temps réel : c'est un ensemble de modules assurant le traitement des interruptions du système et l'activation de programmes correspondants. Sa complexité peut être plus ou moins étendue selon l'importance du système. Il peut s'agir d'un simple traitement d'interruptions avec branchement au sous-programme correspondant comme d'un système permettant d'avoir les programmes à activer sur disque avec gestion du swap.
- la gestion du processus proprement dit : il s'agit d'un ensemble de programmes capables à eux seuls d'assurer le fonctionnement du processus. Cet ensemble est en liaison avec deux sources possibles de dialogue :
  . le processus lui-même : il s'agit de traiter les informations qu'il envoie et d'envoyer les ordres adéquats pour la bonne marche du système,
  . un système de commande : il s'agit d'un ensemble de programmes pouvant se dérouler concurremment avec la gestion du processus et qui assure le dialogue entre l'opérateur et le contrôle du processus, en permettant à l'opérateur de pouvoir agir par un langage simple sur le cours du déroulement du processus.

Un tel système traduit donc les ordres de l'opérateur en séquences d'ordres élémentaires envoyés au contrôle du processus ou en séquences internes modifiant la structure des tables utilisées par le système.

- un système de visualisation : permettant à l'opérateur de lire les divers paramètres du processus ou de visualiser sous une forme quelconque la marche du processus ou les produits obtenus. Un tel système peut très bien se concevoir comme un sous-ensemble du système de commande, auquel on a accès par une clé spéciale.

On a donc le schéma représenté figure 6.

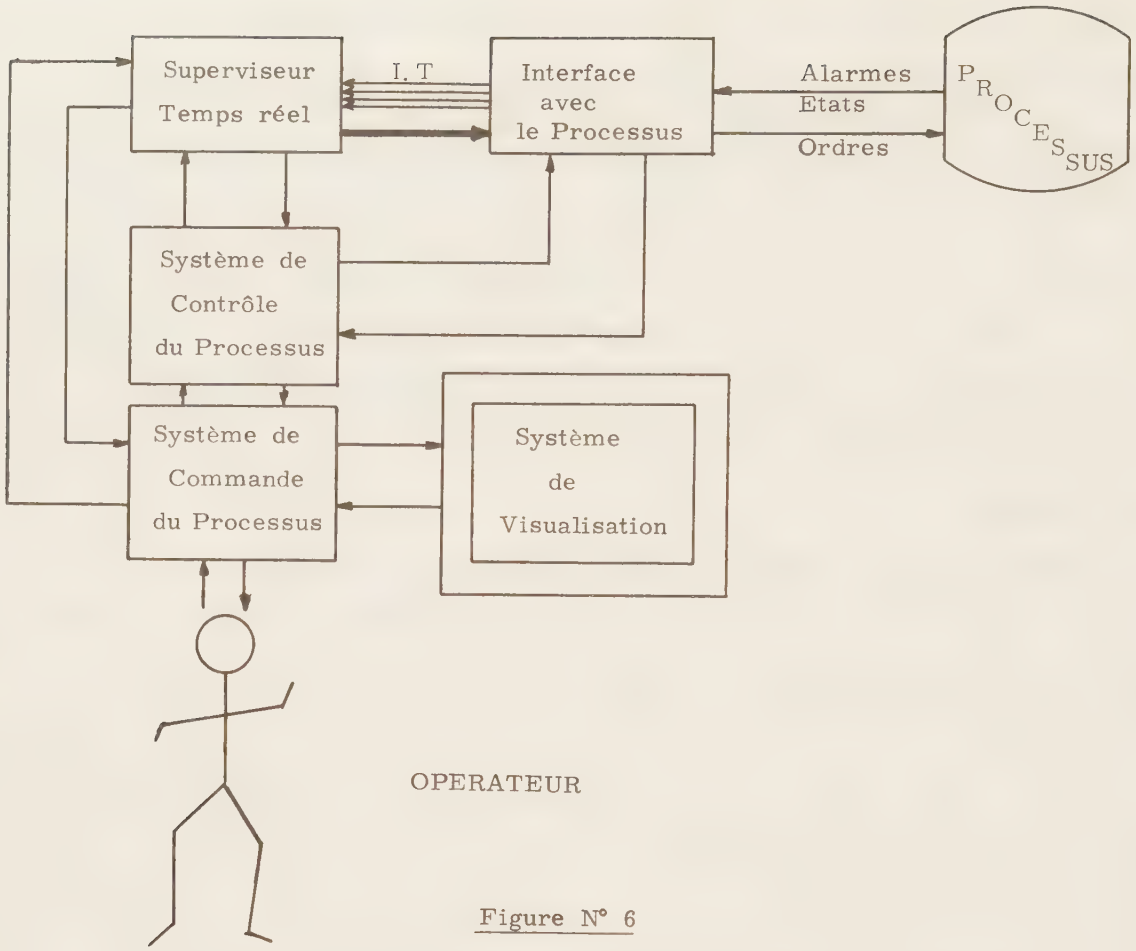


Figure N° 6

Vue d'ensemble d'un Système Informatique contrôlant un Processus

I. 4 - Commandes opérateur

Nous avons donc constaté l'utilité d'un système permettant à l'opérateur de dialoguer avec l'ordinateur, d'où la notion de "commande-opérateur".

On appelle "commande-opérateur" un ensemble de mots-clé et de constantes, soumis à des règles d'écriture bien précises, et qui, une fois analysé et décodé, est susceptible d'avoir une action bien précise sur le système de contrôle du processus, que ce soit pour envoyer des ordres ou acquérir des états de fonctionnement.

On peut envisager deux types de fonctionnement :

- fonctionnement analytique : la commande constitue une phrase d'entrée, contenant :
  . le mot-clé : indiquant quel est le type de l'action que l'on veut voir s'exécuter,
  . des constantes numériques ou alphanumériques : représentant le sous-ensemble de paramètres nécessaires au mot-clé pour l'exploitation correcte de la commande

Dans ce mode de fonctionnement, la phrase est entrée entièrement, puis analysée syntaxiquement afin de séparer les mots-clés des paramètres, enfin exécutée.

- fonctionnement conversationnel : dans ce cas là, la commande se réduit à un seul mot-clé, qui est immédiatement décodé, et provoque un dialogue, avec demandes et réponses, pour

permettre au système d'acquérir les informations nécessaires au déroulement de la commande.

Pour un contrôle de processus, le mode conversationnel semble le plus indiqué. En effet :

- le système questions-réponses permet à tout novice d'utiliser le système après une mise au courant très courte, sinon nulle,
- l'efficacité est augmentée car les risques d'erreur sont faibles, puisque c'est le système qui guide l'opérateur. En cas d'erreur d'entrée d'un paramètre, seul ce paramètre est redemandé,
- la rapidité d'action sur le système est plus grande, car l'identification d'un mot-clé est plus rapide qu'une analyse syntaxique, avec des risques d'erreurs quasi-nuls. C'est intéréssant pour un processus dans le cas d'une alarme, où on doit agir rapidement,
- il semble plus adapté à la notion de modularité, ce qui est intéressant pour l'adjonction ou la modification de commandes.

I.5 - Action des "Commandes Utilisateur"

Une fois la commande et ses divers paramètres entrés, la commande doit alors provoquer une certaine action au niveau du processus, ou du système qui le pilote. Les commandes destinées à un processus peuvent se ranger en trois grandes catégories :

- commandes dynamiques externes : ce sont les commandes qui provoquent une action au niveau du processus lui-même, à travers le système qui le contrôle bien entendu. Ce sont les équivalents des boutons-poussoirs d'un pupitre de contrôle classique du processus,
- commandes dynamiques internes : ce sont les commandes qui n'agissent pas directement sur le processus, mais sur les tables, valeurs de consigne, butées et autres parties software du système de contrôle. Elles peuvent bien entendu affecter le processus lui-même, mais de façon indirecte.
- commandes statiques de visualisation : ces commandes laissent inchangé le processus, ainsi que son système de contrôle. Par contre elles permettent de visualiser sous une forme quelconque les divers états, les diverses variables du système. Elles jouent le rôle des appareils enregistreurs, des écrans de contrôle d'une commande classique de processus. Elles peuvent par exemple permettre l'édition automatique d'un journal de bord à intervalles fixes.

Les commandes dynamiques externes peuvent avoir des conceptions différentes suivant la complexité du système de programmes qui contrôle le déroulement du processus

- soit ce dernier est un ensemble de sous-programmes courts ayant pour but de traiter les interruptions et alarmes du processus et d'envoyer les commandes adéquates. Ceci est surtout valable dans le cas d'un processus ayant un cycle où le matériel impose le programme si l'on peut dire. Dans ce cas, la commande externe consistera essentiellement à des entrées-sorties directes avec le processus, restant au stade de la commande élémentaire citée plus haut.

- dans le cas où le processus n'est pas cyclé, le matériel peut évoluer de façon quelconque, et il y a en général un ou plusieurs programmes capables de piloter la marche du processus pour accomplir une fonction bien déterminée et tout à fait conditionnée par ce programme. Dans ce cas, la commande utilisateur aura surtout une action sur le système de déroulement des programmes contrôlant la marche du processus.

I.6 - Modularité d'un système de contrôle de processus

L'ordinateur a fait son entrée dans les contrôles de processus pour les faire bénéficier de ses possibilités de calcul, ainsi que de sa rapidité. La souplesse d'emploi faisait également partie des avantages espérés d'un tel système. Or, malgré la notion de programme qui indique que la volonté du programmeur est toute puissante, il n'est pas exclu de tomber dans la rigidité par une mauvaise conception du système et de rendre ainsi toute modification irréalisable, tout au moins sans tout reprendre.

C'est pourquoi il est très intéressant d'introduire ici la notion de modularité dans le pilotage des processus, domaine dans lequel les progrès du matériel, les découvertes de nouvelles techniques, imposent des changements et il est intéressant de n'avoir à modifier que quelques instructions, en laissant le reste inchangé.

Il est donc très judicieux d'analyser toutes les phases du déroulement d'une commande et de voir quelles sont celles susceptibles de pouvoir être réutilisées par d'autres commandes ; on doit pouvoir, après cette analyse et ce découpage, disposer d'un ensemble de modules permettant d'exécuter, au prix de quelques instructions supplémentaires peut-être, toutes les commandes envisageables pour le processus étudié.

I.7 - Organisation type d'un système de commande modulaire

Dans ce qui suit, nous allons supposer que nous disposons d'un superviseur permettant d'activer des modules, ces modules pouvant également s'appeler l'un l'autre. Nous supposerons également que le système se veut conversationnel.

L'exécution d'une commande peut se découper en trois phases :

- l'entrée de la commande
- l'analyse et reconnaissance
- l'exécution proprement dite

Au cours de l'exécution d'une commande, on peut s'attendre à avoir :

- des messages demandant l'entrée de paramètres
- des entrées d'informations
- des impressions d'erreur
- des actions spécifiques à la commande

On peut donc déjà prévoir :

- un module d'entrée de la commande
- un module analyse
- un module impression de message

- un module impression d'erreur (faisant appel au module précédent)
- un module entrée d'informations (distinct de l'entrée de la commande)
- un module propre à la commande envisagée

I.7.1 - Module d'entrée de la commande

Nous avons dit qu'il fallait prévoir un module d'entrée de la commande distinct du module entrée informations. Ce n'est pas une condition impérative, mais cela nous parait plus commode pour plusieurs raisons.

- Il est très commode de figer le format d'une commande, en tenant compte de sa représentation en mémoire. Par contre, lors de l'entrée de paramètres, il est vraisemblable que le format sera variable (chiffres, lettres, mélange des deux). Bien sûr, qui peut le plus peut le moins, mais la simplicité gagne à distinguer ces deux types d'entrée.
- très souvent, l'entrée de la commande est provoquée par un appel extérieur de l'opérateur générant une interruption. Ce module est donc directement activé de l'extérieur, alors que l'entrée de paramètres est appelée à partir d'un autre module. Or, le plus souvent, ces deux modes d'appel sont incompatibles dans les superviseurs et l'on est amené à créer un module distinct pour l'entrée de la commande.

I.7.2 - Module analyse

Ce module a pour rôle de reconnaître la commande et de se brancher vers le traitement approprié, qui peut consister à appeler un module propre à la commande.
On peut distinguer :

- les commandes longues : qui nécessitent beaucoup d'instructions avec des appels d'autres modules,
- les commandes brèves : qui sont le plus souvent des modifications d'état du système informatique et qui ne nécessitent pas de longues séquences.

Dans ce dernier cas, il est intéressant d'effectuer le traitement complet de la commande à l'intérieur du module analyse, quitte à allonger un peu celui-ci ; on gagne en place mémoire car l'introduction d'un nouveau module prend de la place dans les tables du superviseur.

Un autre problème se pose : celui de la sécurité de fonctionnement du processus. Il se peut que certaines commandes soient dangereuses à dérouler vu l'état du système: soit il y a risque de "casser" le système informatique proprement dit, soit risque de détériorer le matériel. Il est donc nécessaire de créer une variable "état du système", initialisée au lancement, et remise à jour lors de l'émission de chaque commande par un automate.

Cet automate doit normalement avoir sa place dans le module analyse, où soit il bloque la commande en cas de danger, soit il la laisse passer en remettant à jour l'état système.

I.7.3 - Autres modules

Ces autres modules ne présentent pas de caractéristiques spéciales et sont essentiellement fonction, soit du système d'Entrées/Sorties, soit de la commande à effectuer.

I. 7. 4 - Schéma type d'un système de commande

L'organisation type d'un système de commande est donnée dans la figure n° 7.

I. 8 - Introduction de la Multiprogrammation

En général, vu la rapidité de traitement des évènements extérieurs par l'ordinateur, il ne serait pas rentable d'utiliser un ordinateur pour un seul processus, à moins que ce dernier ne soit d'une complexité extrême. On connecte donc plusieurs processus sur le même ordinateur, chaque processus possédant un terminal pour la liaison avec l'opérateur.

Cela ne devrait théoriquement rien compliquer : le superviseur temps réel est susceptible de gérer un grand nombre de modules, et il suffit de concevoir autant de modules que nécessaire pour que tous les processus puissent fonctionner ensemble.

Or, il ne faut pas perdre de vue :

- que l'ordinateur utilisé n'est pas un gros ordinateur, donc que la taille de sa mémoire rapide est limitée
- que si l'on veut un temps de réponse bref, tous les modules de tous les processus doivent pouvoir être présents en même temps en mémoire rapide.
- que plus il y a de modules, plus les tables du superviseur sont importantes, donc la place disponible en mémoire centrale pour les modules diminue d'autant.

On arrive donc très rapidement au maximum des possibilités du matériel. Or, la conception modulaire du système de commande, permet déjà un gain de place très important. En effet, certains modules peuvent être communs à tous les processus : il suffit de leur préciser le numéro de processus, qui peut alors servir au module pour reconnaître sur quelles variables travailler. Il va de soi qu'un tel module doit être alors réentrant, ou que la réentrance soit assurée par le système.

Si nous considérons le cas du système de commande, il est bien entendu que les divers processus peuvent avoir des "jeux" de commande différents : ce n'est pas très gênant, il suffit de marquer les commandes pour savoir quelles sont celles autorisées pour tel ou tel système.

Dans la pratique, les processus connectés sont du même type et le jeu de commande est le même pour tous. Donc, en particulier, le module d'analyse peut être commun.

Dans le cas le plus complexe, le déroulement du processus est géré par un programme, écrit dans un langage de niveau quelconque. Chaque processus possède son propre programme, et tous doivent pouvoir se dérouler quasi-simultanément aux yeux des opérateurs de chaque processus. Ces programmes pouvant être plus ou moins lourds, il n'est pas question de les avoir tous présents en même temps. On utilise donc une technique de partage de la mémoire entre les divers utilisateurs : la plus répandue est celle du time-sharing. Cette méthode consiste à allouer la zone programme de la mémoire centrale une fraction de temps T à chaque utilisateur. Les tops d'horloge sont envoyés par un organe externe.

En fait, il arrive souvent qu'entre l'envoi d'un ordre au processus et la réponse de ce dernier, il s'écoule un temps assez long, pouvant aller jusqu'à des dizaines de minutes. Dans ce cas, le programme gérant le processus n'a rien à faire, sinon attendre, et

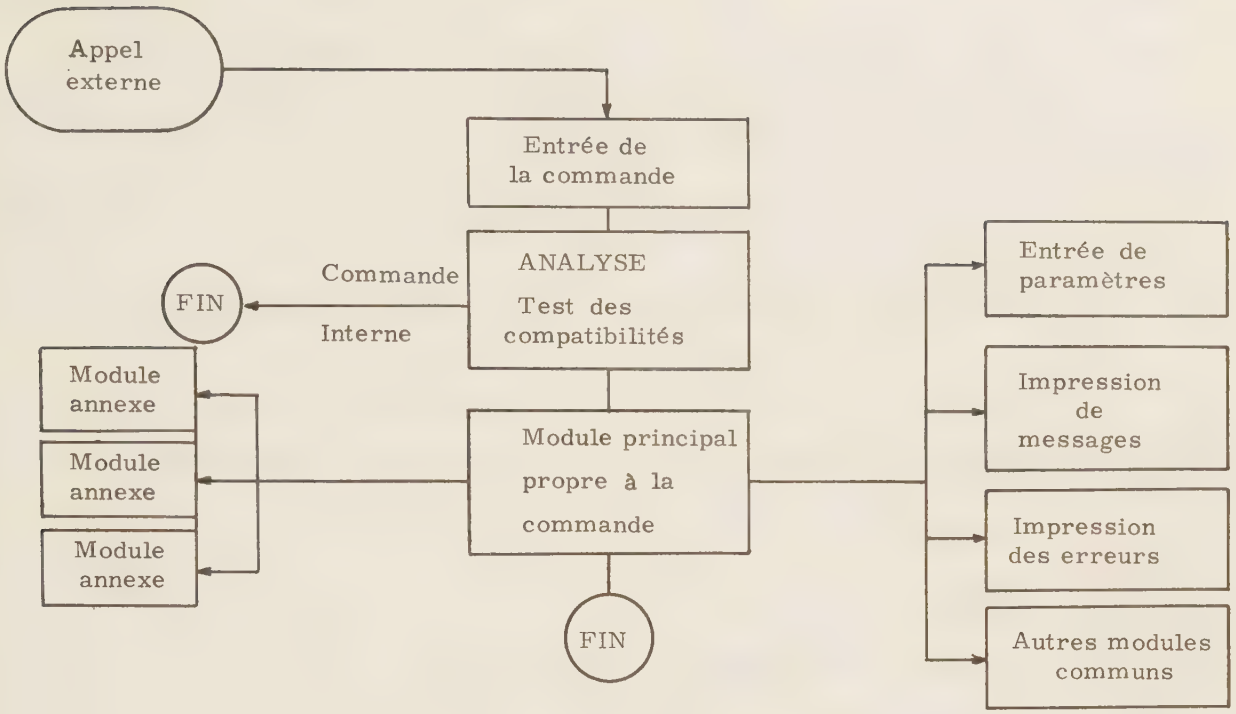


Figure N° 7

Exemple d'un Système modulaire de Commande de Processus

il ne sert à rien de le laisser actif. Il est particulièrement judicieux d'utiliser alors un time-sharing optimisé comme suit : lorsque, durant sa phase d'activité pour le système de time-sharing, un programme passe en attente pour une raison quelconque, il est immédiatement désactivé et on passe à l'utilisateur suivant. Lors des cycles suivants, l'utilisateur sera toujours considéré comme inactif tant que la réponse du processus ne sera pas parvenue. Ce système de partage est très intéressant car, il permet de diminuer le nombre d'utilisateurs vus par le système, donc d'améliorer les temps de réponse.

On pourrait même pratiquement se passer des impulsions de hâchage en considérant que le passage en attente longue d'un programme utilisateur fait basculer sur l'utilisateur suivant, et ainsi de suite. Cependant, on est obligé de recourir aux impulsions de time-sharing pour empêcher qu'un programme ayant peu d'attente ne monopolise trop longtemps l'ordinateur.

On peut voir sur un exemple simple le gain d'une telle méthode, sans rentrer dans tous les détails.

Supposons que l'on ait n utilisateurs déroulant le même programme ; supposons que, pour le déroulement de ce programme, on ait besoin de p cycles, pas forcément utilisés dans toute leur durée t ; soit s le temps système pour passer d'un utilisateur à l'autre.

Supposons enfin, que le coefficient d'utilisation de la mémoire centrale soit k (c'est-à-dire le rapport du temps effectivement passé en ordinateur, sur la durée totale du programme, attente comprise).

En moyenne, à chaque cycle, un utilisateur ne reste en mémoire que : kt secondes (t = période des impulsions).

Lorsque tous les programmes sont achevés, il se sera écoulé :

$$n.p.\ kt \text{ secondes (n. p cycles en tout)}$$
$$+\ n.p.\ s \text{ (temps système) secondes}$$

Si l'utilisateur était seul, il lui faudrait p cycles, soit $T = pt$. L'attente est donc de :

$$A = n.p.kt + n.p.s - pt = pt\,(nk-1) + nps$$

L'attente relative a pour valeur :

$$a = \frac{A}{T} = \frac{pt\,(nk-1) + nps}{pt} = (nk-1) + n.\frac{s}{t}$$

$$\boxed{a = (nk-1) + n.\frac{S}{t}}$$

On retombe sur une formule classique du temps partagé (voir cours de M. LECARME - "Système exploitation d'ordinateurs"), où l'on voit que, bien entendu, l'attente est d'autant plus grande qu'il y a plus d'utilisateurs. Cependant, on remarque que, dans le premier facteur, le nombre n d'utilisateurs est multiplié par un coefficient k, inférieur à 1, donc que pour le système le nombre d'utilisateurs est diminué dans ce rapport. Notons que ceci est conforme au bon sens puisque un utilisateur en attente longue est ignoré par le système, sauf pour le temps de passage de l'un à l'autre (on constate bien que k n'intervient pas dans le second membre).

Cette relation pourrait également laisser croire que si k est très faible, on gagne du temps et que l'attente devient négative, ce qui ne signifie rien. En fait, la durée totale, même en supposant tous les autres utilisateurs inactifs, est au moins supérieure ou égale à la durée du porgramme utilisateur tout seul. Donc :

$$n.p.kt \geq pt \Longrightarrow nk \geq 1$$

On voit donc que la multiprogrammation est très intéressante dans un système de pilotage de processus, car les programmes de gestion du processus étant souvent en attente d'une réponse de celui-ci, on peut avoir un nombre d'utilisateurs plus important, sans altérer le temps de réponse du système.

II - SUPERVISEUR R.T.D.M.S

Préliminaires

Après ces considérations générales sur le contrôle de processus par systèmes informatiques, nous allons maintenant rentrer dans la description effective du système réalisé pour l'ILL. Ce chapitre veut donner une description sommaire du Superviseur R.T.D.M.S. de Télémécanique contrôlant la marche du système. Nous mettons bien l'accent sur le fait que ce produit est la propriété de la Société Télémécanique et, qu'à ce titre, nous ne pouvons entrer dans les détails, mais seulement définir les traits généraux et surtout un certain vocabulaire facilitant la compréhension des parties suivantes.

Nous signalerons cependant qu'ayant eu la version d'essai de ce superviseur, nous avons eu de nombreux problèmes qui ont nécessité un examen approfondi de R.T.D.M.S., et souvent des modifications du source existant. En particulier, nous avons complètement modifié le traitement des interruptions en provenance des périphériques.

II.1 - Généralités

Le superviseur R.T.D.M.S. (Real Time Disk Monitor System) est un ensemble de programmes permettant le déroulement "simultané" de modules appelés tâches selon une certaine priorité, les tâches pouvant résider en mémoire centrale ou sur disque. On lui a adjoint un moniteur d'E/S MONES permettant aux tâches d'effectuer des échanges d'information avec les périphériques standards. Un système de fichier SYSFIC non évolutif, permet le rangement de données sur disque par adressage symbolique, avec noms et articles.

Les diverses tâches se répartissent en quatre catégories :

- Classe 1 : directement activées par les interruptions externes
- Classe 2 : activées à partir d'un classe 1 ou d'un autre classe 2
- Classe 3 : sous-programme réentrant rattaché à un classe 2
- Classe 4 : module de R.T.D.M.S. accessible par les autres tâches et considéré par le superviseur comme une interruption externe.

Le superviseur possède une pile d'interruption où sont sauvegardés les principaux registres lorsqu'une interruption externe arrive. Mis à part les programmes de classe 1, qui sont des sous-programmes de traitement d'interruption pouvant être écrits par l'utilisateur, ce dernier écrit ses programmes spécifiques sous forme de tâches de classe 2 ou 3. Chaque classe 2 est identifié par un nombre de 12 bits baptisé PTU (pointeur tache utilisateur) et qui sert d'index pour les tables de R.T.D.M.S. Les tâches ayant les PTU les plus faibles sont les plus prioritaires. La tâche de PTU zéro est spéciale : elle est systématiquement appelée lors du lancement du superviseur, ou en cas d'erreur système. C'est donc dans cette tâche que l'utilisateur doit faire l'initialisation de son système.

Les tâches de classe 2 peuvent être swappables : un module spécial est chargé de les charger en mémoire centrale ou de les renvoyer sur le disque.

Le superviseur R.T.D.M.S. gère la barre de verrouillage, et la positionne au pied du dernier classe 2. Au-delà, se trouve le background qui est le programme ayant la plus faible priorité. Le franchissement de la barre de verrouillage provoque des alarmes qui

sont traitées par le Superviseur, lequel peut autoriser la tentative faite si elle est répertoriée dans ses tables.

Le coeur du superviseur est constitué par le "schéduleur", module qui scrute l'état de tous les classes 2 dans l'ordre de leur priorité et décide de les activer ou non.

II.2 - Tâches de classe 1

Ce sont des séquences de traitement des interruptions externes. Elles utilisent des modules de classe 4 du superviseur, effectuent les travaux nécessités à la réception de l'interruption et redonnent le contrôle à la fin au Schéduleur par le module RPI.

Certaines parties de ces séquences peuvent être masquées afin de s'affranchir des autres interruptions plus prioritaires : bien entendu, elles le sont le moins souvent possible afin de ne pas diminuer les performances du système.

Les interruptions en provenance des bacs d'Entrées/Sorties demandent à être aiguillées vers le Moniteur d'E/S MONES, lequel redonne ensuite le contrôle au schéduleur.

La séquence classique pour un classe 1 est donc la suivante :

- sauvegarde des registres du programme interrompu
- analyse et identification de la source d'appel
- traitement approprié (éventuellement activation de tâches de classe 2)
- retour au schéduleur par RPI ou par MONES puis RPI si interruption d'E/S .

II.3 - Tâches de classe 2

Ce sont des séquences d'instructions rebouclées sur elles-mêmes, dont la longueur est un multiple de 64 instructions. Elles s'implantent en mémoire à des adresses octales multiples de 100 (conséquence de la longueur). Ces impératifs sont nécessités par le système de SWAP.

Les 8 premiers mots d'une tâche sont réservés pour le système :

- 5 mots de sauvegarde des registres en cas d'interruption
- 1 mot où est rangé un argument lors de l'activation de la tâche
- éventuellement 2 mots de compte-rendu de Swap si la tâche est sur disque.

La tâche se termine par un retour au superviseur par le module RPT. L'instruction qui suit le branchement vers RPT dans la tâche est une instruction de branchement vers le début de la tâche, pour une activation ultérieure, car la sauvegarde du tampon de la tâche après un RPT pointe sur l'instruction qui suit, comme dans le cas d'une interruption externe (nous avons dit plus haut qu'un classe 4, ici RPT, se comporte vis à vis des tâches utilisateur comme une interruption externe).

Une tâche de classe 2 est dite bloquée, lorsqu'elle possède dans les tables du superviseur un anneau d'appel où sont enregistrées les demandes consécutives d'activation de cette tâche. Une tâche non bloquée ne peut être réactivée une 2ème fois, si le 2ème appel intervient alors qu'elle se déroule suite à un premier appel.

Lorsqu'un utilisateur veut faire dérouler un programme, autre qu'un traitement d'interruptions, ce dernier doit être écrit sous la forme d'une ou plusieurs tâches de classe 2.

Lorsque dans un classe 2, on veut faire des opérations d'Entrées/Sorties, il faut obligatoirement passer par le moniteur d'Entrées/Sorties MONES.

II.4 - Appel (ou activation) de tâches

Nous avons déjà dit que les tâches de classe 1 étaient directement activées par les interruptions externes. Il suffit donc de charger leurs adresses dans les mémoires d'interruption.

Les tâches de classe 2 peuvent être activées :

- à partir de classe 1 par le module APP
- à partir de classe 2 par le module APT

Lorsqu'on active une tâche, on indique :

- la nature de l'appel
- son PTU
- l'argument à lui transmettre.

L'appel peut être de deux sortes :

- séquentiel : la tâche appelante est mise en attente, et n'est réactivée que sur la fin de la tâche appelée,
- non séquentiel : la tâche appelante n'est pas mise en attente, mais ne sera réactivée que s'il n'y a pas d'autres tâches plus prioritaires activables. Ainsi, un appel non séquentiel à une tâche plus prioritaire, est équivalent, du point de vue séquence de déroulement, à un appel séquentiel.

Il est bien évident qu'un appel à partir d'un classe 1 par le module APP est un appel non séquentiel, le module se contentant de mettre la tâche dans un état activable pour le schéduleur, et rendant la main aussitôt au classe 1.

Il n'est pas possible de passer un argument lors d'un appel par un classe 1. De même, on ne peut pas passer zéro (tous les bits à zéro) comme argument à partir d'un classe 2.

Lorsqu'une tâche est bloquée, les divers appels sont mémorisés dans un anneau. On mémorise :

- le PTU de l'appelant si l'appel est séquentiel
- l'argument d'appel.

Pour ces raisons, il n'est pas possible d'appeler une tâche bloquée à partir d'un classe 1.

Une tâche bloquée n'a pas besoin d'être réentrante, la réentrance étant assurée par le système. Prenons un exemple :

- soit une tâche A, activée par une interruption (1), (à partir d'un classe 1) faisant appel à une tâche B bloquée avec un argument a,
- soit une tâche C, plus prioritaire activée par une interruption (2) faisant appel également à B avec un argument c.

On a la séquence représentée sur la figure n° 8.

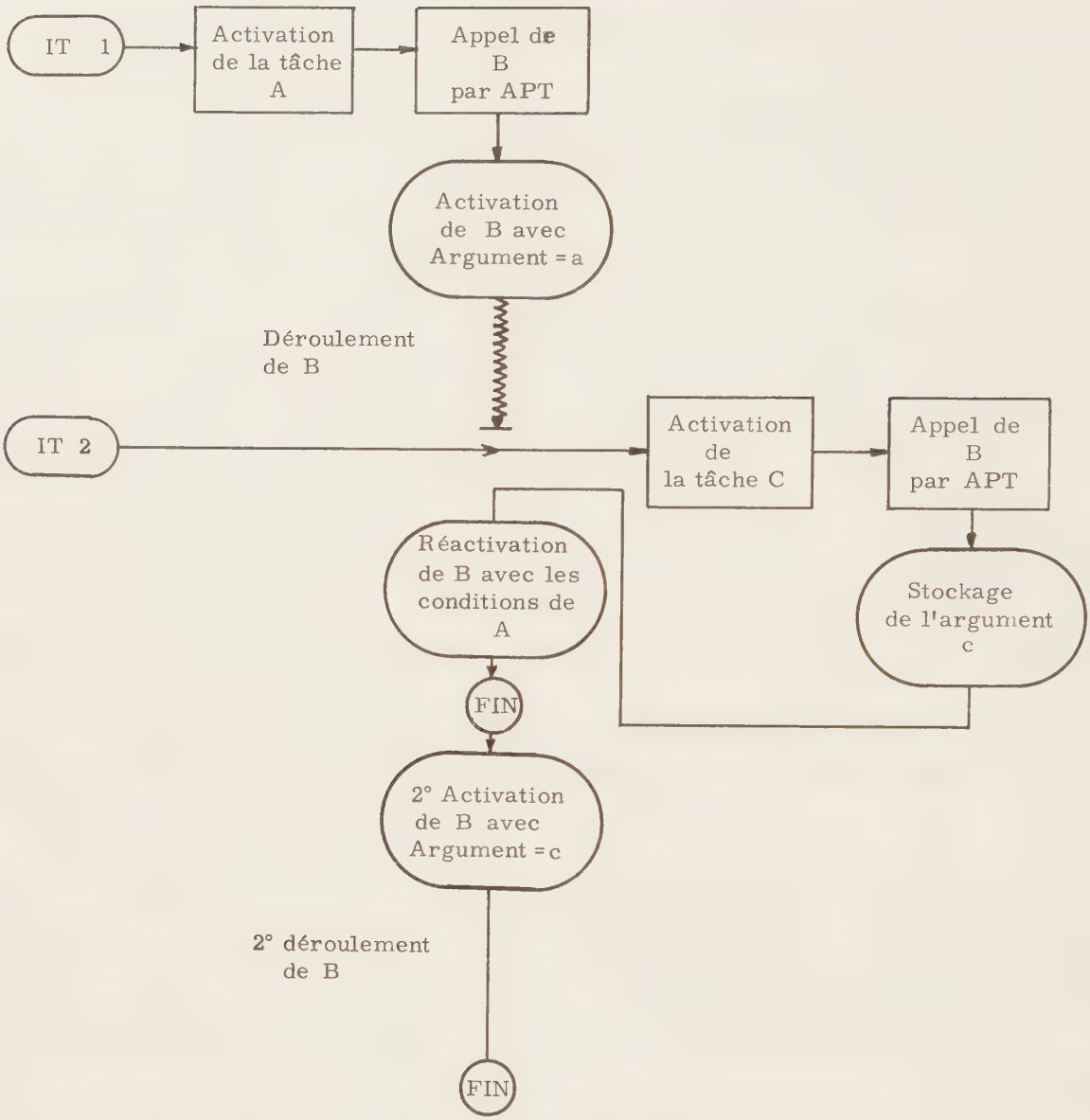


Figure N° 8

Séquence de déroulement d'une tâche bloquée interrompue pour un second appel

On voit donc qu'un classe 2 interrompu, est ensuite réactivé avec les mêmes conditions et qu'il ne se déroulera à nouveau pour un 2ème appel (pris en compte dans l'anneau) que lorsque le 1er appel sera terminé.

Lorsqu'on a deux appels consécutifs sur une tâche non bloquée, seul le 1er appel est pris en compte, le second est ignoré.

II.5 - Tâches SWAPPABLES

Les tâches de classe 2 peuvent être swappables, et donc sont implantées sur disque et amenées en mémoire centrale au moment de leur exécution. Le superviseur possède donc dans ses tables leur adresse sur disque.

Lorsqu'une tâche swappable doit être activée, le superviseur

- vérifie si la place nécessaire en mémoire est libre ; sinon, si la tâche actuellement présente est moins prioritaire elle est renvoyée sur disque. Autrement on attend sa fin.
- lance la demande de swap au module de swap,
- réactive d'autres tâches, s'il y en a, durant le swap,
- sur interruption de fin de swap, active la tâche alors présente.

Lorsqu'une tâche swappable est interrompue par une tâche plus prioritaire, s'implantant au même endroit, elle est recopiée dans son état actuel sur disque et est considérée comme non présente. Lors de sa réactivation, elle sera swappée à nouveau en mémoire centrale, et reprise là où elle avait été interrompue.

En particulier, lors d'un appel non séquentiel à une tâche plus prioritaire swappable, l'appel n'est pas tout à fait séquentiel, comme on l'a dit plus haut : en effet, la tâche appelante peut être réactivée durant le temps du swapp de la tâche appelée.

II.6 - Entrées-Sorties par MONES

Le moniteur d'E/S MONES gère tous les échanges des classes 2 avec les périphériques standards.

Lorsque le périphérique fonctionne avec les E/S programmés, MONES simule le fonctionnement en mode canal. Dans le cas où le périphérique fonctionne en mode canal, MONES sert de simple interface avec RTDMS afin d'assurer un déroulement correct des tâches appelantes.

Donc, le classe 2 désirant effectuer une entrée/sortie fait une demande d'échange à MONES par un branchement particulier.

Cet appel à MONES peut être :

- séquentiel : la tâche appelante est mise en attente et ne sera réactivée que sur la fin de l'échange demandé,
- non séquentiel : MONES, après avoir lancé l'échange, rend le contrôle à la tâche appelante (s'il n'y en a pas de plus prioritaire activable entre temps) et échange et classe 2 se déroulent simultanément,
- non séquentiel avec réveil : le fonctionnement est identique au précédent, sinon qu'à un certain moment le classe 2 a besoin d'être sûr que l'échange est terminé.
  Dans ce cas, il fait appel au module REVEIL qui
  . met la tâche en attente si l'échange n'est pas terminé,
  . rend le contrôle au classe 2 si l'échange est terminé.
  Ce mode d'appel est donc un appel séquentiel différé.

Lors de l'appel à MONES, la tâche de classe 2 transmet toutes les informations nécessaires à l'échange en fournissant au moniteur l'adresse d'une table d'échange.

Cette table d'échange peut être directement utilisée par le canal automatique lorsque le périphérique fonctionne en mode canal.

II.7 - Adressage symbolique des périphériques

En plus de la table d'échange, le classe 2 fournit également au moniteur d'Entrées/Sorties :

- l'unité symbolique de l'échange (adresse symbolique du périphérique)
- le travail à faire

MONES possède un système d'adressage symbolique des périphériques dont nous allons donner quelques caractéristiques. A chaque périphérique est attachée une table d'Unité Périphérique (Table UP en abrégé) qui contient tous les renseignements relatifs au périphérique proprement dit, c'est-à-dire :

- témoins indiquant l'état du périphérique et ses caractéristiques (ex. Télétype, fonctionnement en mode anal, etc....)
- numéro de l'unité symbolique pointant sur la table UP (voir plus loin)
- adresse du périphérique dans le bac d'E/S et adresse du bac
- adresse du driver en entretien
- adresse du driver en défaut ou initialisation

Les "drivers" cités plus haut sont des petits modules chargés d'assurer le déroulement correct de l'échange et spécifiques des périphériques pour lesquels ils ont été écrits.

Vers ces tables UP peuvent pointer une ou plusieurs tables d'Unité Symbolique. Ces tables (en abrégé table d'US) servent de tampon pour conserver les divers renseignements relatifs à une demande d'échange. De plus, certaines mémoires servent de mémoire de travail à MONES. Les adresses de des diverses tables sont rangées dans une table appelée TBS, et on désigne par unité Symbolique le nombre qui sert d'index pour aller chercher la bonne table d'US dans la TBS.

Lorsqu'un échange est lancé sur une US, celle-ci est dite occupée jusqu'à la fin de l'échange. Toute demande d'échange sur la même US sera refusée si celle-ci est déjà occupée. Par contre, nous avons précisé que plusieurs Tables d'US pouvaient pointer vers la même table UP. Donc, si un échange est en cours avec un périphérique sur une US donnée, on peut demander un autre échange sur une autre US pointant sur la même table UP. L'échange ne sera pas lancé immédiatement puisque le périphérique est en cours, mais l'appel est mémorisé. On range dans une pile le numéro de l'US en attente, et à la fin du premier échange, on examine la pile, et l'échange sur le périphérique sera lancé par l'intermédiaire de la 2è US mise en attente préalablement.

L'ensemble des Tables d'US et des tables d'UP constitue donc une file d'attente à structure horizontale pour les échanges avec périphériques, les caractéristiques de l'échange étant mémorisés dans les tables d'US. De même les tables d'US et d'UP servent de tables de réentrance pour MONES, qui peut ainsi gérer plusieurs périphériques simultanément.

Schématiquement, on a la disposition représentée à la figure n° 9.

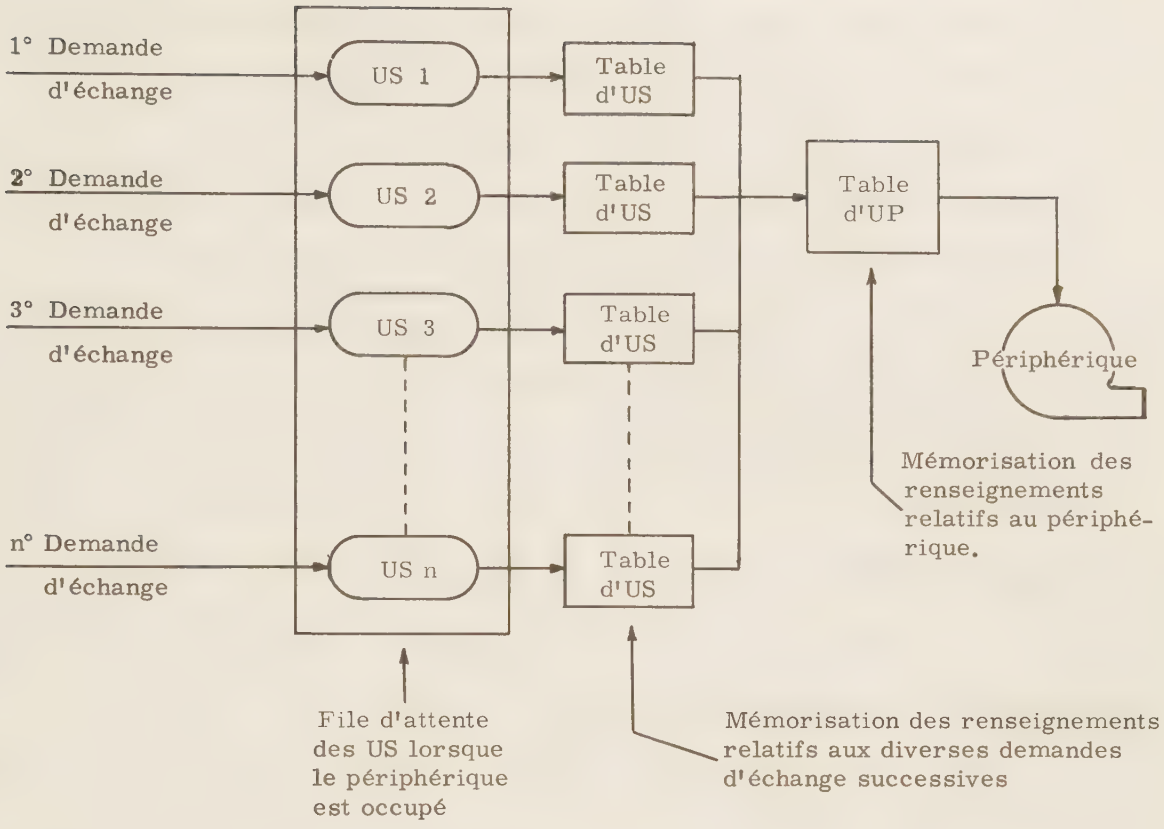


Figure N° 9

Schéma Synoptique des tables d'US et d'UP

II.8 - Travaux effectués par MONES - Comptes Rendus

MONES peut accomplir deux sortes de travaux :

- travaux immédiats : ce sont des travaux ne nécessitant pas d'échanges d'information avec le périphérique et donc rapides. Il s'agit surtout de modifier ou de lire les paramètres internes au système d'adressage symbolique (Ex. demande de mot état périphérique, demande de compte-rendu, demande de tuer un échange etc....)
- demandes de travail réel : ce sont des demandes nécessitant un ou plusieurs échanges avec le périphérique, donc de longue durée : par exemple, demande de transmission de mot de commande, demande de transmission de données.

Dans tous les cas, MONES fournit un compte-rendu sur deux mots, indiquant si l'échange s'est bien passé, sinon pourquoi il s'est mal passé. Ce compte-rendu est chargé à une adresse fournie dans la table d'échange.

En fait, MONES fournit deux comptes-rendus successifs, le second venant écraser le premier :

- compte-rendu relatif à la requête
- compte-rendu relatif à l'échange proprement dit.

II.9 - Précautions à prendre dans le cas de tâches SWAPPABLES

Tout d'abord, précisons que, dans le cas d'une tâche swappable, l'appel à MONES, outre les divers types énoncés plus haut peut être :

- Swappable : la tâche peut être swappée out pendant l'échange,
- non swappable : la tâche reste présente en mémoire pendant toute la durée de l'échange. Une tâche s'implantant sur elle, même plus prioritaire, ne pourra pas être activée tant que l'échange n'est pas terminé.

Donc, il faut faire attention essentiellement à deux choses :

- au buffer d'entrée ou de sortie : s'il est dans la tâche elle-même, l'échange doit être non-swappable sous peine de fonctionnement aléatoire. Sinon le mettre en résident,
- au Compte-Rendu : comme pour le buffer, prendre des précautions, car la tâche risque de ne plus être là lorsque le compte-rendu arrive, et celui-ci risque donc de détruire certaines instructions d'une autre tâche implantée au même endroit.

II.10 - Système de gestion de fichiers disque "SYSFIC"

Le module SYSFIC de RTDMS est un module de classe 4 qui permet la gestion de fichiers disques avec possibilité d'adressage symbolique. Ce module utilise le moniteur d'E/S MONES pour les échanges avec le disque.

Le système n'est pas dynamique en ce qui concerne les fichiers, mais peut l'être pour ce qui est des articles.

Un fichier disque est un ensemble d'articles : chaque fichier peut comporter jusqu'à 512 articles.

Un article est constitué par un ensemble de mots de 20 bits, dont le nombre ne peut excéder 4096.

Le nombre des fichiers est quelconque, mais doit être obligatoirement figé à la création du système, ce à cause des tables utilisées par sysfic, qui ne peuvent être générées dynamiquement.

De même, le nombre maximal d'articles autorisé pour un fichier doit être fixé à la configuration du système.

Chaque fichier peut être appelé par un nom de 3 caractères alphanumériques. L'appel des articles peut se faire soit par nom, soit par numéro, qui est leur rang dans le fichier. Ce deuxième mode est plus rapide.

La correspondance entre les noms est assurée par deux tables :

- table des Fichiers TDF qui assure le lien entre le nom et l'adresse réelle sur disque,
- table des Articles TDA qui contient tous les renseignements quant aux caractéristiques du fichier.

L'appel au module SYSFIC se fait en désignant un numéro qui est le numéro de l'UAF sur laquelle on veut travailler. Une UAF est une table qui assure la réentrance de SYSFIC et, est un peu équivalente aux tables d'US pour MONES.

Bien entendu les appels à SYSFIC peuvent être rejetés si l'on ne tombe pas dans les conditions déterminées à la génération, et SYSFIC transmet un compte-rendu indi-

quant la nature de l'erreur commise. SYSFIC est donc un système assez souple et assez commode d'emploi, bien que n'étant pas dynamique. Remarquons que l'on peut créer quand même dynamiquement des articles, mais non des fichiers. Certaines particularités de SYSFIC, utilisées par la suite, seront détaillées lorsque le moment sera venu.

II.11 - Conclusions

L'ensemble RTDMS + MONES + SYSFIC constitue donc un superviseur temps réel permettant au programmeur d'écrire facilement ses programmes sous forme de tâches, avec tous les impératifs dûs au fonctionnement en temps réel. La possibilité d'avoir des fichiers symboliques sur disque permet une grande souplesse et une économie assez conséquente de place en mémoire centrale, tout en n'occasionnant pas de pertes de temps trop considérables, le temps passé pour les E/S étant utilisé pour exécuter les instructions des autres programmes actifs.

Bien entendu, nous n'avons décrit ici que le strict minimum nécessaire à la compréhension de ce qui va suivre.

III - VUE D'ENSEMBLE DU SYSTEME SOFTWARE DE CARINE

III.1 - Généralités

Nous avons déjà exposé dans un chapitre précédent quelle était la structure du système informatique chargé de gérer les expériences du système CARINE, et pour quelles raisons on en était arrivé à une telle organisation.

Nous rappellerons que ce système est composé de trois sous ensembles :

- le système de gestion du FORTRAN temps réel SYSFOR
- le système de gestion des Entrées/Sorties avec le CAMAC MONAC
- le système de dialogue et de contrôle du déroulement des programmes COSYNUS

Ces trois ensembles sont étroitement liés entre eux et ont été développés en structure modulaire permettant des adjonctions ou modifications aisées. Ils sont tous trois placés sous le contrôle du superviseur temps réel RTDMS.

Chacun de ces systèmes donnera lieu à une description détaillée dans les pages qui vont suivre.

III.2 - Description de l'implantation en mémoire du système

Dans la partie qui décrivait le hardware installé sur CARINE, nous avons précisé que la mémoire centrale disponible comportait 8 blocs (de 0 à 7) de 4K mots de 19 bits.

Les blocs 0 et 1 contiennent les instructions et constantes de RTDMS, MONES et de l'interface FORES entre le FORTRAN et MONES.

Le bloc 2 contient l'interpréteur FORTRAN.

Le bloc 3 contient l'arithmétique flottante, la gestion des bibliothèques ainsi que le système de bandes magnétiques.

Le bloc 4 contient le moniteur CAMAC MONAC ainsi que le sous-programme GESCAM s'occupant de la gestion des appels FORTRAN au CAMAC.

Le bloc 5 contient les constantes résidentes de COSYNUS et SYSFOR, les tables de MONAC, le buffer des bandes magnétiques et la zone de SWAP commune où se swappent les tâches système.

Les blocs 6 et 7 contiennent le programme FORTRAN de l'utilisateur en cours, ou le Background.

Les figures n° 10, 11, 12 des pages suivantes donnent les implantations mémoire, ainsi que les implantations disque des divers composants du système.

Toutes les adresses indiquées sont en octal.

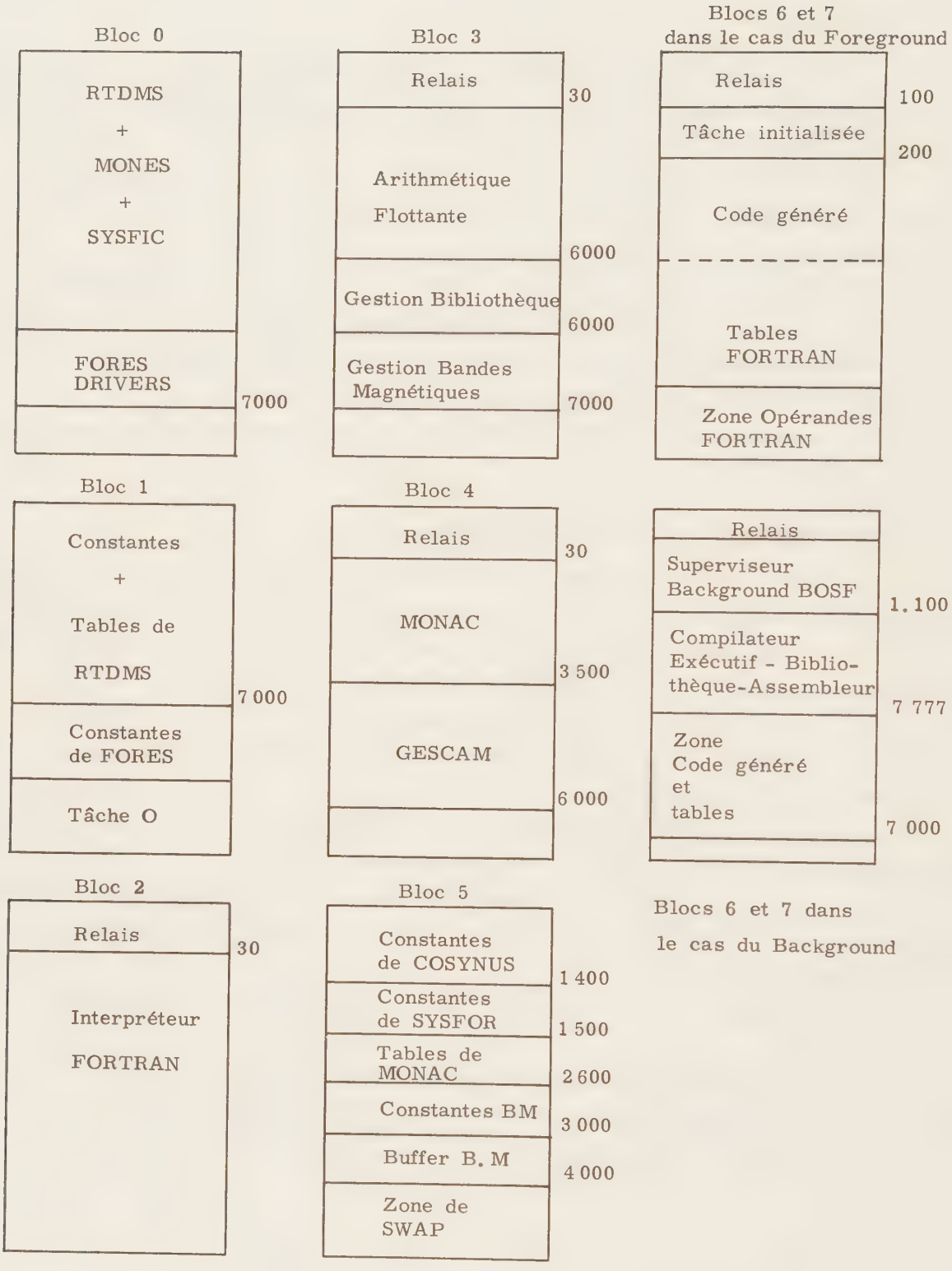


Figure N° 10

Implantation en Mémoire Centrale

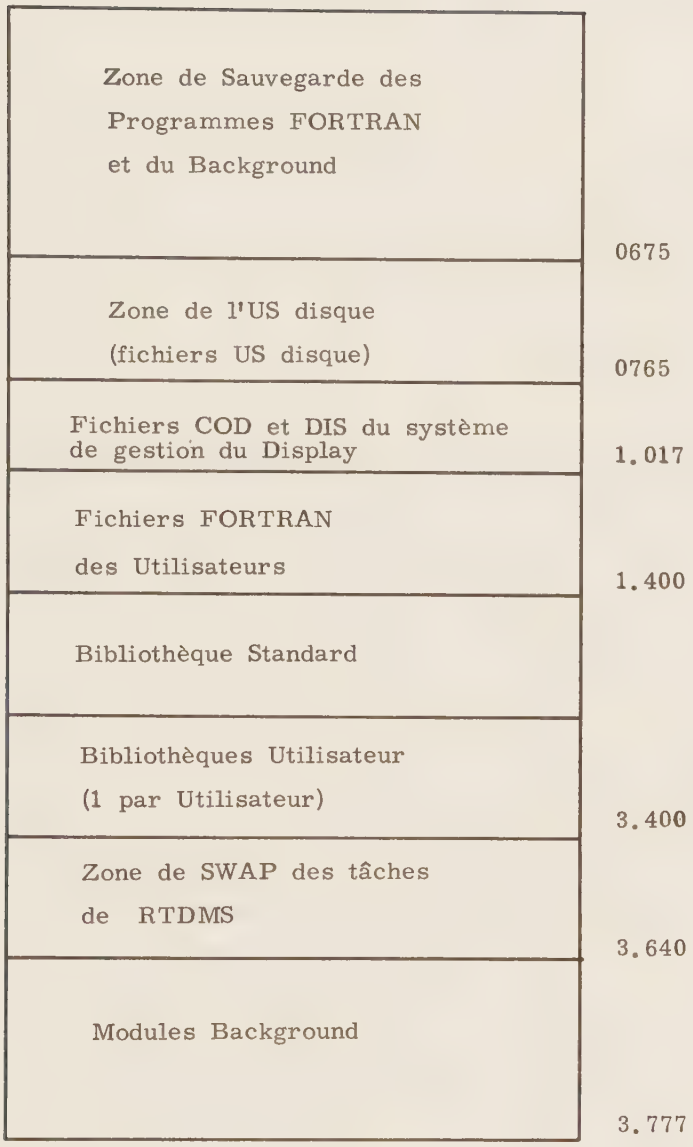


Figure N° 11

Implantation sur Disque du Système

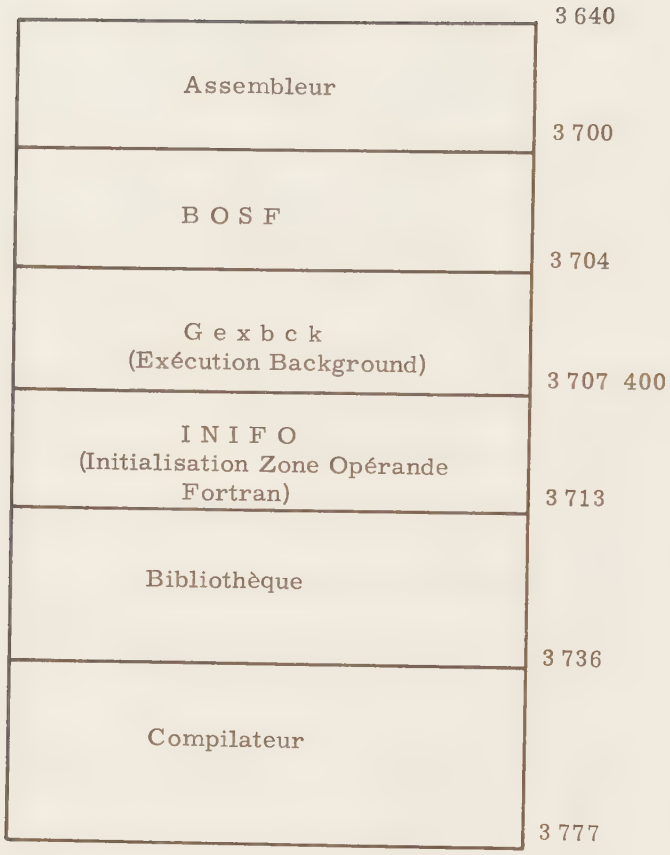


Figure N° 12

Détail de l'Implantation Disque des Modules Background

IV - SYSTEME DE COMMANDE COSYNUS

Préliminaires

Dans ce chapitre, nous allons décrire la partie du système qui a été conçue et réalisée par l'auteur du présent rapport.

Aussi, nous nous sommes permis de donner de nombreux détails, ainsi que toutes les contraintes qui nous ont conduit à réaliser le système tel qu'il est défini dans les pages qui suivent.

IV.1 - Généralités

Le système COSYNUS, associé au Superviseur Temps Réel RTDMS, est un ensemble de programmes permettant à plusieurs expérimentateurs de contrôler en temps réel le déroulement de leur expérience. Dans la version actuelle, ce système permet de gérer

- 6 TTY utilisateurs
- 1 TTY système
- 1 TTY Background

Sa conception modulaire permet l'adjonction facile de commandes ou sous-commandes à condition de respecter certaines règles d'écriture. L'utilisation de tables de correspondance TTY → Utilisateur, de tables de PTU, de flags de protection, lui assurent une souplesse et une sûreté de fonctionnement remarquables. Pour ce faire, les drivers et sous-programmes d'interruption du RTDMS Standard ont été modifiés, car la conception de l'Appel Superviseur et des Codes d'Arrêt dans le Superviseur peut introduire des aléas de fonctionnement.

Le découpage en tâches pouvant se swapper l'une sur l'autre, permet l'utilisation d'une faible zone mémoire. L'emploi de tâches séparées pour les entrées-sorties permet de diminuer le nombre de tâches nécessaires sur disque, tout en accroissant considérablement les performances du système du point de vue du temps d'attente pour l'utilisateur.

IV.2 - Cahier des charges primitif - Performances obtenues

Le système de commande pour les expériences couplées sur CARINE devait présenter les avantages suivants :

- conception modulaire permettant l'adjonction facile de commandes
- place mémoire la plus faible possible
- encombrement disque minimal
- sûreté de fonctionnement même dans les cas les plus tangents
- temps d'attente minimal pour l'utilisateur
- détection préalable de toutes les erreurs et correction facile et rapide
- souplesse d'emploi
- fonctionnement nécessitant le moins d'intervention possible de la part de l'utilisateur tout en lui laissant une grande liberté d'utilisation
- réinitialisations et détection des pannes Hardware si possible.

Les points les plus délicats, qui allaient déterminer la conception même du système étaient donc les suivants :

- place mémoire et place disque optimales
- temps d'attente minimal pour chaque utilisateur

Tous ces points sont intimement liés car :

- on peut diminuer la place mémoire en utilisant des tâches se swappant l'une sur l'autre, mais on augmente la place disque et on augmente le temps d'attente dans le cas d'entrées-sorties longues déclarées non-swappables, si on ne veut utiliser des Buffers résidents trop nombreux ou si on utilise dans MONES des tables chaînées,
- on peut diminuer l'attente en utilisant une tâche particularisée pour chaque utilisateur, mais on augmente alors la place disque et la place mémoire, car, à un même instant, on peut avoir toutes les tâches présentes en mémoire, sinon le gain de temps sur l'attente serait illusoire,
- on peut réduire la place disque en utilisant des tâches communes à tous les utilisateurs, mais on augmente considérablement les temps de réponse du système aux yeux de l'utilisateur, surtout dans le cas d'entrées longues.

D'autre part, le problème de la sûreté de fonctionnement était très crucial, car il faut prévoir toutes les réactions de l'utilisateur face au système. En particulier, le problème de l'Appel Superviseur TTY, qui est l'unique moyen pour alerter le système, devait présenter à la fois la sûreté et la souplesse indispensables.

Le système COSYNUS, réalisé en tenant compte des impératifs qui précèdent, semble offrir à l'utilisateur les performances souhaitées :

- souplesse d'emploi par un jeu de commandes standards permettant le contrôle facile de l'expérience grâce à un dialogue précis mais bref avec le système,
- temps de réponse rapide (maximum 3 à 4 s dans les cas les plus défavorables) dûs à la conception des tâches d'E/S pour un utilisateur donné,
- sûreté de fonctionnement grâce à l'utilisation de flags de protection sur l'Appel Superviseur et d'Etats utilisateurs interdisant certaines commandes dans les cas dangereux,
- place mémoire propre au système COSYNUS minimale comme suit :
  - . résident : 640 mots
  - . zone de swap propre à COSYNUS : 384 mots

  Soit 1K au total
- place disque pas trop importante ne dépassant pas 8K avec :
  - . gestion des TTY
  - . gestion des E/S
  - . gestion du FORTRAN Temps réel et du Time-Sharing
  - . gestion des erreurs
  - . gestion TTY Système
- modularité grâce au découpage réalisé permettant une réutilisation facile des tâches déjà écrites et la création de nouvelles commandes.

IV.3 - Traitement spécial de l'appel superviseur

IV.3.1. Le traitement de l'Appel Superviseur est un sous-programme très important de COSYNUS, car c'est le seul moyen d'alerter l'ordinateur. De plus, la conception même de RTDMS avec MONES et le système d'appel des tâches incite à prendre les précautions maximales afin d'éviter tout appel intempestif pouvant casser le système.

Tout d'abord, faisons un bref rappel sur la philosophie de l'Appel Superviseur telle qu'elle est conçue dans RTDMS + MONES. L'Appel Superviseur y est considéré comme un défaut (conséquence du Hardware). D'où les conséquences suivantes :

- si l'appel monte en dehors d'un échange sur la TTY considérée, on risque de casser le système, car on fait un compte-rendu à une adresse inexistante puisqu'il n'y a pas eu de tables d'échange,
- si l'échange est en cours, on tue l'échange ; on dispose alors d'un compte-rendu permettant de savoir qu'il y a eu Appel Superviseur.

Le problème le plus gênant réside dans le fait que, si l'Appel Superviseur monte simultanément avec une demande de libération de la TTY (RDEF et ROCC positionnés) au cours d'un échange, le traitement fait par le module ANAL de RTDMS peut ignorer le bit de ROCC monté avec le défaut et empêcher ainsi un échange de se terminer dans certains cas (par ex. plusieurs US sur la même UP).

Il est donc apparu nécessaire de changer les traitements des IT du bloc E/S programmé afin de remédier à tous ces inconvénients.

La philosophie adoptée pour le traitement de l'Appel Superviseur est la suivante :

- l'Appel Superviseur
  . en dehors du déroulement d'une tâche
  . en l'absence d'E/S sur la TTY considérée
  active une tâche dont le PTU est dans une mémoire attachée au niveau de la TTY. Dans cette mémoire peuvent se trouver 3 PTU :
    - PTU de la tâche TRCP entrant les commandes principales
    - PTU de la tâche TACLE entrant le mot de passe si le système est verrouillé
    - PTU de la tâche Superviseur VISU si l'on est en VISU
- si on est en cours de déroulement d'une tâche, et en dehors d'un échange sur la TTY considérée, l'Appel Superviseur est inactif.
- si on est en cours de déroulement d'une tâche, et pendant un échange sur la TTY
  1) on tue l'échange sur la TTY
  2) si la TTY était occupée par COSYNUS, on positionne un flag et on termine
  3) si la TTY était occupée par un autre système que COSYNUS, on appelle le PTU qui est dans la mémoire liée à l'Appel Superviseur.

Dans le cas n° 2, le flag positionné est testé dans la tâche COSYNUS et c'est la tâche elle-même qui décide ce qu'elle a à faire sur cet Appel.

Avec cette philosophie, et en utilisant des flags de protection, des Appels successifs rejetés ne craignent en aucun cas de "casser" le système (sans cela, on risque en effet de rappeler une tâche non bloquée, provoquant des erreurs systèmes, ou de bloquer des Tâches ou des US attachées à la TTY).

IV.3.2 - Autres problèmes liés à l'Appel Superviseur

Un autre problème primordial se posait : comment tuer un échange sur Appel Superviseur ?

On peut essayer de considérer l'Appel comme un défaut et de se rebrancher dans MONES, mais cela présente des aléas si l'échange est en cours pour les raisons citées plus haut (simultanéité de ROCC et RDEF).

La solution adoptée a été la suivante :

- si le périphérique est en cours (UP occupé) lors de l'Appel Superviseur
  1) on range l'adresse d'un module spécial dans le mot 7 de la Table d'UP (adresse de Driver en entretien)
  2) on envoie le caractère NULL (tous les bits à zéro) sur la TTY considérée, et on attend l'UR en retour)
  3) sur l'UR on se branche sur le module spécial qui appelle ou non le PTU attaché à la TTY et on fait un compte-rendu similaire à un code d'arrêt, ce qui termine correctement l'échange.

Avec ce traitement, l'Appel Superviseur tue tous les échanges en cours sur la TTY sans rien casser dans le système.

IV.3.3 - Réalisation pratique des sécurités

A chaque utilisateur sont associées deux mémoires :

- la première, liée à l'appel superviseur, comporte deux bits indiquant :
  qu'une tâche est en cours,
  qu'il y a eu Appel Superviseur pendant l'échange,
- la seconde contient l'état de l'utilisateur qui permet de refuser certaines commandes dans les cas dangereux.

IV.4 - Problème des entrées-sorties

Les Entrées/Sorties sont gérées par le moniteur MONES inclu dans RTDMS. Ce moniteur a dû être modifié à cause de son mauvais fonctionnement dans certains cas particuliers. Sans entrer dans le détail, les points suivants ont dû être modifiés :

- problème des Comptes-Rendus
- problème des Codes d'Arrêt
- problème traitement des défauts Canal
- adjonction d'un Système ATTACHE-DETACHE

Une fois ces problèmes résolus, restait celui du temps de réponse pour les entrées-sorties vu du côté utilisateur.

IV.4.1 - Données de départ

Supposant parfaitement connu du lecteur le fonctionnement de MONES, il est bon de rappeler les faits suivants :

- lorsque l'on fait des chaînages de Tables d'échange, la tâche doit être déclarée non swappable pendant la durée de l'échange, sinon on risque de ne pas trouver la table d'échange

suivante si la tâche a été swappée out entre temps,

- si le buffer d'entrée ou de sortie est non résident, la tâche doit être déclarée non swappable pendant l'échange pour les mêmes raisons,
- les entrées doivent être séquentielles, sauf cas très spéciaux, car on a besoin des données en entrée pour continuer le programme.

D'autre part, on aurait pu envisager de faire une tâche par utilisateur, toutes les tâches pouvant être présentes ensemble en mémoire rapide. Cette solution augmenterait considérablement la place disque et la place nécessaire en mémoire rapide.

- une solution possible était de créer une tâche commune bloquée pouvant être appelée par tous les utilisateurs. Mais, si cette tâche comporte des entrées longues, en admettant qu'il y ait 6 appels à cette tâche, il faut que le dernier appelant attende 5 déroulements de la tâche avant d'avoir le contrôle, ce qui peut représenter des dizaines de minutes d'attente.

IV.4.2 - Solution choisie

D'où s'imposait la solution de faire des tâches spécialisées d'E/S par utilisateur, se swappant l'une sur l'autre. Restait le problème des Entrées nécessitant un appel séquentiel de la tâche d'E/S et présentant donc les inconvénients déjà cités. On a donc décidé que l'on appellerait non séquentiellement la tâche d'E/S et que, lorsque celle-ci se terminerait, elle appellerait la tâche appelante avec un argument spécial indiquant que l'entrée s'est déroulée.

Donc, chaque utilisateur a à sa disposition, pour les E/S :

- une tâche ECL qui entre un certain nombre de caractères jusqu'au RC avec possibilité d'annulation par " ← ",
- une tâche IMPRI qui imprime les messages contenus dans un BUFFER résident,
- une tâche TRCP qui entre les 3 caractères d'une commande,
- une tâche TACLE qui entre un mot de passe.

On a créé les tâches TACLE et TRCP distinctes de ECL car ces tâches sont directement activées par Appel Superviseur, et les conditions d'utilisation de RTDMS impliquent qu'on ne peut faire un APP sur une tâche bloquée.

Une description détaillée de ces tâches sera donnée ultérieurement.

IV.5 - Description d'une commande COSYNUS

IV.5.1 - Définition

Une commande COSYNUS est un ensemble de 3 caractères alphanumériques, qui décodés et identifiés, activent une ou plusieurs tâches pour effectuer un travail déterminé.

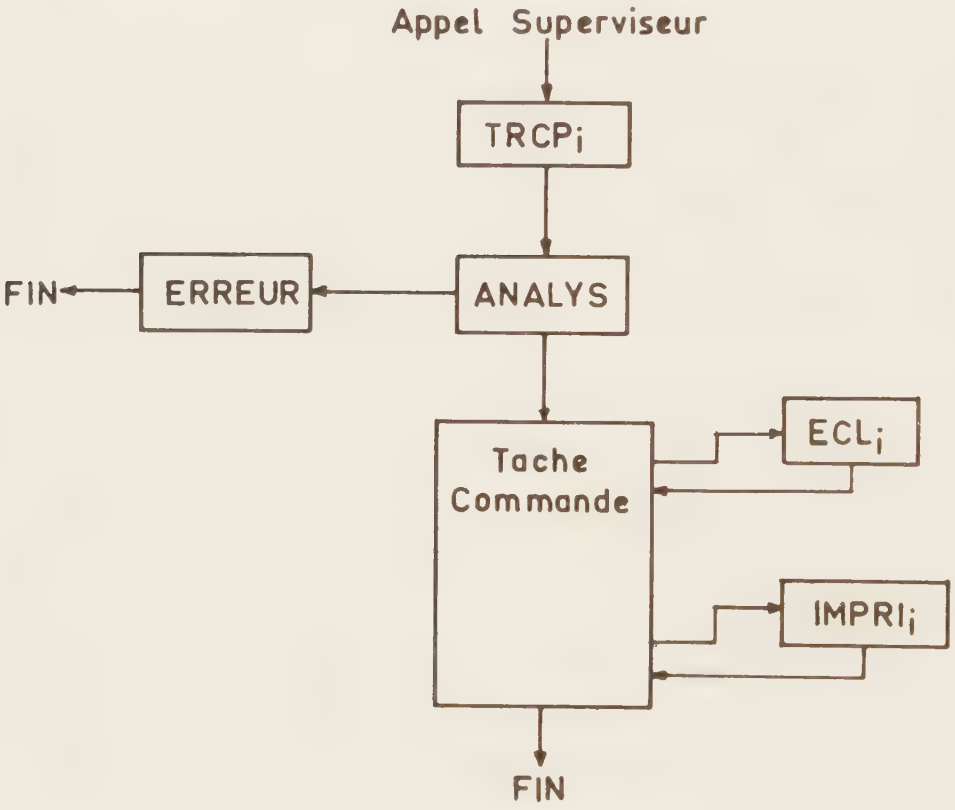
IV.5.2 - Tâches systèmes utilisées par la commande

- la commande est entrée par la tâche TRCP qui la range en 3C/M dans le premier mot de la zone Buffer de l'utilisateur,

- elle est analysée par ANALYS qui effectue les vérifications de compatibilité et active la tâche correspondante avec le bon argument,
- pour les entrées, la tâche commande utilise la tâche ECL de l'utilisateur considéré et se termine en attendant d'être rappelée par la fin de l'Entrée Clavier,
- pour les sorties, elle utilise la tâche IMPRI qui permet l'impression d'un message avec éventuellement l'heure et la mise hors tension de la TTY.

La tâche commande est une tâche commune à tous les utilisateurs, le numéro de l'utilisateur lui est donné en argument.

Le déroulement type d'une commande est donc le suivant :



IV.5.3 - Mise en place nouvelle commande

Pour mettre en place une nouvelle commande, il faut :

- rajouter les éléments nécessaires dans ANALYS
- incorporer la nouvelle tâche dans la version de RTDMS, c'est-à-dire définir :
  - . PTU
  - . TES
  - . TNI
  - . TRE
  - . TAD

IV.5.4 - Cas spéciaux

Dans tout ce qui précède était sous-entendu qu'une commande est associée à une tâche dont la longueur dépend de l'importance de la commande. Pour certaines commandes, LOK par exemple, qui est un simple verrouillage, le traitement très court est fait dans ANALYS.

Pour les commandes FORTRAN, au contraire, ANALYS a un rôle très bref, l'analyse de compatibilité étant faite dans une autre tâche (JTLANS) pour ne pas avoir la même séquence dans plusieurs tâches.

IV.5.5 - Règles à respecter dans l'écriture des tâches commandes

Au début de la tâche TRCP qui entre la commande, le système COSYNUS :

1) Positionne un flag indiquant le début d'une commande,
2) Attache la TTY utilisateur au système.

Donc, en fin de commande il faut :

1) RAZ le flag correspondant,
2) Détacher la TTY

IV.5.6 - Implantation des tâches commandes

Les tâches commandes, activées à partir d'ANALYS s'implantent dans la Zone de SWAP banalisée (7200 à la fin du bloc 5). Elles doivent donc faire moins de 6 secteurs, soit 384 mots.

IV.6 - Description des "tâches systèmes" de COSYNUS

Par "tâches Système", on désigne les tâches nécessaires au déroulement des commandes, et constituant le noyau de COSYNUS, les commandes étant des modules pouvant se rajouter à volonté, dans la limite de la place disponible.

Ces tâches sont les suivantes :

- TRCP qui permet l'entrée de la commande,
- TACLE qui permet le déverrouillage du système par l'entrée d'un mot de passe,
- ECL qui permet une entrée clavier de 20 caractères Maxi,
- IMPRI qui permet l'impression d'un message
- ERREUR qui permet l'impression d'un message d'erreur
- ANALYS qui reconnaît la commande, vérifie les compatibilités et active la tâche correspondante.

Il est à noter que les 4 premières tâches sont répétées autant de fois qu'il y a d'utilisateurs, alors que les deux dernières tâches sont communes à tous les utilisateurs.

IV.6.1 - Section TAAS - Traitement Appel Superviseur - DRIVER TTY-LR-PR-LC

Cette section est une tâche particulière, puisqu'il s'agit d'un Classe 1, c'est-à-dire qu'il est directement activé par une IT externe et constitue le traitement proprement dit de l'IT.

Ce module permet le branchement vers MONES des IT en provenance du bloc d'E/S programmées et remplace les modules correspondants de la version standard de RTDMS.

Un bref rappel du traitement des IT dans RTDMS : sur l'IT, on se branche sur une séquence qui :

- fait un RESER pour sauvegarder les registres
- fait un rangement du tampon dans PILIN
- se branche sur ANAL pour l'analyse des sous-niveaux
- termine par un RPI pour faire retomber le niveau et redonner le contrôle au schéduleur.

Dans ANAL, il y a une table de branchement avec 15 adresses correspondant aux 15 sous-niveaux du bac d'E/S. Ces adresses sont précisément les adresses des modules remplacées par TAAS.

Un module type est de la forme :

```
RE      SVE
CE    MSQIT        Masquage IT
RX    I  NB        Sauvegarde de l'index
CXV       + POSUP  POSUP = position de l'UP correspondante dans la table d'UP TBP
IRV   SCRUT2+ 2
```

Il apparaît donc que, si on a 15 périphériques, on devrait avoir 15 modules. Or, ces modules sont identiques, mise à part la valeur de POSUP qui change pour chaque périphérique. On a donc décidé que :

- TBP représenterait la position des périphériques dans le bac d'E/S pour ce qui est des E/S programmées,
- de calculer POSUP et de faire donc un module commun à tous les périphériques fonctionnant en mode programmé,
- d'ajouter à ce module un traitement spécial pour l'Appel Superviseur.

Rappelons, en effet, que l'Appel Superviseur n'est pas considéré comme un Appel, mais comme un défaut, et qu'en ce sens, il tue un échange s'il est en cours et est ineffectif s'il n'y a pas d'échange ne cours.

Donc, dans TAAS il y a deux parties :

- une partie qui traite les UR de libération (RDEF = 0) des périphériques et qui est strictement équivalente aux modules standards de RTDMS,
- une partie qui traite les UR défauts et isole le cas des Appels Superviseurs de la TTY.

Dans le traitement spécial de l'Appel Superviseur proprement dit, on fait le traitement suivant :

- s'il n'y a pas de tâche en cours (Bit 11 de TABOU = 0) on met en appel la tâche TRCP correspondante et on termine sans repasser dans MONES,
- s'il y a une tâche en cours en E/S sur la TTY :
  - on tue l'échange en cours
  - si c'est une US COSYNUS qui est en cours, on positionne un flag (bit 0 de TABOU) et on termine

. si c'est une autre US, on appelle la tâche TRCP correspondante.

IV.6.2 - Tâche TRCP

Cette tâche, associée à un utilisateur, permet l'entrée d'une commande à partir de la TTY. Cette tâche, qui effectue une entrée clavier, constitue une exception au système d'E/S de COSYNUS, car elle n'utilise pas les tâches ECL et IMPRI.

En effet, cette tâche est appelée à partir d'un classe 1 (section TAAS) par le module APP qui ne permet pas la transmission d'argument. De plus, pour que le système d'E/S soit intéressant, il faudrait que cette tâche soit une tâche commune à tous les utilisateurs donc bloquée si on veut prendre en compte tous les appels. Or, ceci est impossible car, on ne peut pas utiliser le module APP pour appeler une tâche bloquée.

On a donc créé une tâche TRCP particularisée par utilisateur, qui fait les E/S par appel direct à MONES, sans passer par les tâches ECL et IMPRI. De plus, cette tâche s'implante en mémoire au même endroit que les tâches ECL et IMPRI. On ne perd donc pas de place mémoire, et on gagne en rapidité et sécurité de fonctionnement.

La tâche TRCP positionne au début TABOU qui indique qu'une commande, enchaînant une série de tâches, va être lancée, et qui est utilisée par TAAS. Ensuite, elle imprime le message

OK n

numéro utilisateur

et demande une entrée clavier en faisant une demande àMONES avec ATTACHE. Autrement dit, la TTY utilisatrice est attachée à l'US COSYNUS dès que TRCP est lancée. Elle ne sera détachée que lorsque la commande se terminera.

Après entrée de la commande, la tâche TRCP appelle la tâche ANALYS, qui analyse la commande, en lui passant en argument le numéro de l'utilisateur. Ensuite TRCP se termine.

Il est à noter qu'après la demande à MONES, il y a test du bit 0 de TABOU pour savoir s'il y a eu appel superviseur pendant l'échange. Si OUI, il y a rebouclage au début de la tâche et le message OK n est réimprimé sur la TTY.

IV.6.3 - Tâche TACLE

Cette tâche est analogue à la tâche TRCP. Elle est activée comme elle par l'Appel Superviseur dans le cas où l'on a verrouillé le système.

Pour les mêmes raisons que celles déjà indiquées, il y a une tâche TACLE par utilisateur. Cette tâche s'implante à la place de TRCP.

La tâche TACLE n'attache pas la TTY aux US COSYNUS. Il y a impression du message.

KIVALA

et attente d'un mot de passe de 3 lettres.

Ce mot de passe est comparé au code entré

- s'il est correct, il y a déverrouillage du système et activation de la tâche TRCP correspondante

- sinon on laisse le système verrouillé et on appelle la tâche IMPRI pour mettre la TTY hors tension.

Le système reste verrouillé tant que le bon mot de passe n'est pas transmis.

IV.6.4 - Tâche ECL

Cette tâche est spécialisée dans l'entrée clavier TTY d'un nombre maximal de 20 caractères. L'entrée est terminée par la frappe du caractère RC.

Cette tâche travaille avec comme argument, le PTU à rappeler lorsque le caractère RC est décodé. De plus, si le bit 0 de l'argument n'est pas positionné, il y a conversion en 3C/Mots suivant le standard FORTRAN T 2000, où le blanc est considéré comme séparateur. Dans ce cas, le caractère RC n'est pas rangé, mais la mémoire suivante du buffer est remise à zéro pour indiquer la fin de l'entrée.

La conversion 1 C/m en 3 C/Mot est faite par une tâche spéciale CONVER commune à tous les utilisateurs.

Lorsque ECL se termine, elle rappelle la tâche dont le PTU lui a été transmis en argument. Elle passe en argument

- cadré en 11, le numéro de l'utilisateur
- bit 0 = 1 pour indiquer que l'entrée clavier s'est déroulée correctement
- bit 1 = 1 s'il y a eu Appel Superviseur pendant l'entrée clavier
- bit 2 = 1 si l'appel à MONES a été refusé .

La demande à MONES est faite avec arrêt sur table de codes.

Deux codes d'arrêt sont utilisés :

- le retour chariot RC
- la flêche " ← ".

Après la demande à MONES, il y a analyse du code qui a provoqué la fin de l'entrée,

- si c'est un RC, on termine normalement, avec conversion ou non
- si c'est une flèche, on refait la demande à MONES et on provoque une nouvelle entrée clavier, qui doit annuler la précédente.

IV.6.5 - Tâche CONVER

Cette tâche est utilisée par toutes les tâches ECL pour convertir du 1 C/M en 3 C/Mot.

Le blanc (espace) est considéré comme séparateur. S'il y a plusieurs espaces consécutifs, seul le premier est pris en compte.

En fin de conversion, on remet à zéro la mémoire qui suit le dernier mot significatif pour l'entrée considérée.

Exemples de rangement :

- TITU est rangé en | T | I | T | | | | U |
- TOTITE est rangé en | T | O | T | | I | T | E |
- RA∨LE∨BOL est rangé en | | R | A | | | L | E | | B | O | L |
- Z est rangé en | | | Z |

IV.6.6 - Tâche IMPRI

Cette tâche est spécialisée dans l'impression d'un message sur la TTY utilisateur. Il y a autant de tâches que d'utilisateurs. Elle s'implante en mémoire au même endroit que les tâches TRCP, TACLE et ECL correspondantes.

La tâche IMPRI fait une demande à MONES avec table de codes d'arrêt pour imprimer un certain nombre de mots (passé en argument) rangés en 2 C/Mot dans le buffer de l'utilisateur.

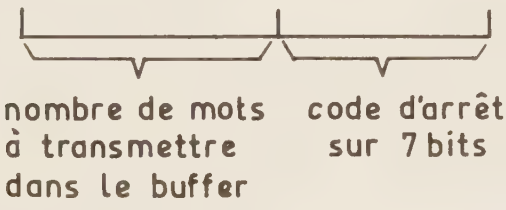
Il y a ensuite une série de chaînages sur :

- impression de deux espaces
- impression de l'heure suivie de RC LF
- mise hors tension de la TTY

Le code d'arrêt sur 7 bits est passé en argument.

- si c'est un espace : on n'imprime que le message contenu dans le buffer,
- si c'est un LF : on imprime en plus l'heure,
- si c'est un zéro (NULL) : on met ensuite la TTY hors tension

L'argument transmis à IMPRI est le suivant :



* Si l'argument transmis est nul, il y a mise hors tension de la TTY tout simplement.

* Remarques sur IMPRI

- IMPRI commence par l'impression systématique d'un RC LF
- Comme dans IMPRI on utilise certains caractères comme code d'arrêt et que le code d'arrêt passé en argument est sur 7 bits, si l'on veut transmettre ces caractères particuliers sans arrêter l'échange, il faut forcer le bit de parité à 1, en sorte que la disjonction avec le code d'arrêt soit différente de 0.

C'est ainsi que, dans le message RC LF du début d'IMPRI, le code LF est représenté par 2120 au lieu de 0120 pour ne pas provoquer l'arrêt du message.

- IMPRI fait un appel séquentiel non swappable à MONES. De plus, IMPRI est plus prioritaire que ECL et s'implante au même endroit. Donc, dans le cas classique où l'on pose une question et où l'on attend la réponse, on peut faire deux appels successifs non séquentiels à IMPRI et ECL. ECL ne pourra se dérouler que lorsqu'IMPRI est terminée.

IV.6.7 - Tâche ERREUR

Cette tâche permet l'impression d'un message sur la TTY utilisateur sous la forme :

ERR ∀ G ∀ 05

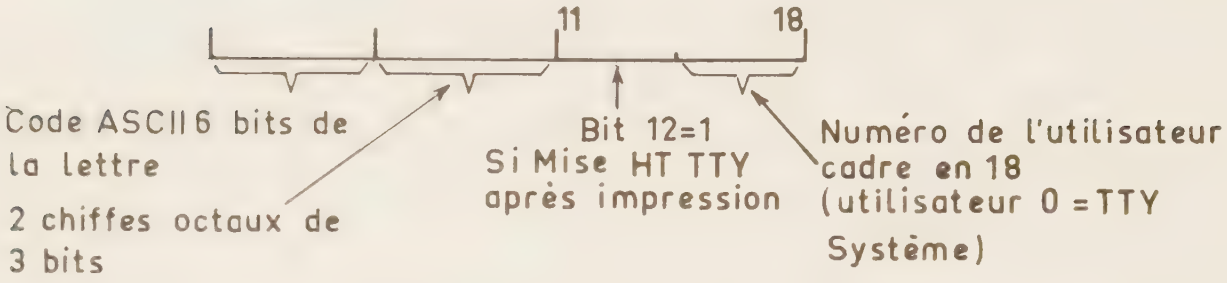
↑ une lettre

↖ 2 chiffres octaux

Elle est commune à tous les utilisateurs.

Elle utilise les tâches IMPRI.

L'argument transmis à ERREUR est le suivant :



ERREUR, après décodage et réarrangement de l'argument, fait appel à la tâche IMPRI correspondant à l'utilisateur et provoque l'impression du message d'erreur ainsi que de l'heure.

IV.6.8 - Section SPREMP

Le module SPREMP est un classe 3 réentrant résident (réentrant pour 7 utilisateurs) qui permet de garnir la zone BUFFER de l'utilisateur avec un message se trouvant dans la tâche, avant d'appeler la tâche IMPRI.

La séquence d'appel est :

CX Numéro Utilisateur

Dans A, (compte de mots à transmettre)

Dans B, adresse de la zone où l'on trouve le Message dans la tâche.

IV.6.9 - Tâche ANALYS

Cette tâche, commune à tous les utilisateurs, fait l'analyse de la commande et le traitement correspondant.

On lui passe en argument le numéro de l'utilisateur cadré en 11.

Elle comporte une table de codes et une table de branchements correspondants.

On distingue en gros trois sortes de commandes :

- les commandes spéciales système (LOK, BAF, DIS)
- les commandes FORTRAN
- les commandes banales, hors FORTRAN.

Pour les commandes banales, le traitement standard est :

CX Numéro utilisateur

IRV ESBON <Hors Fortran ?>

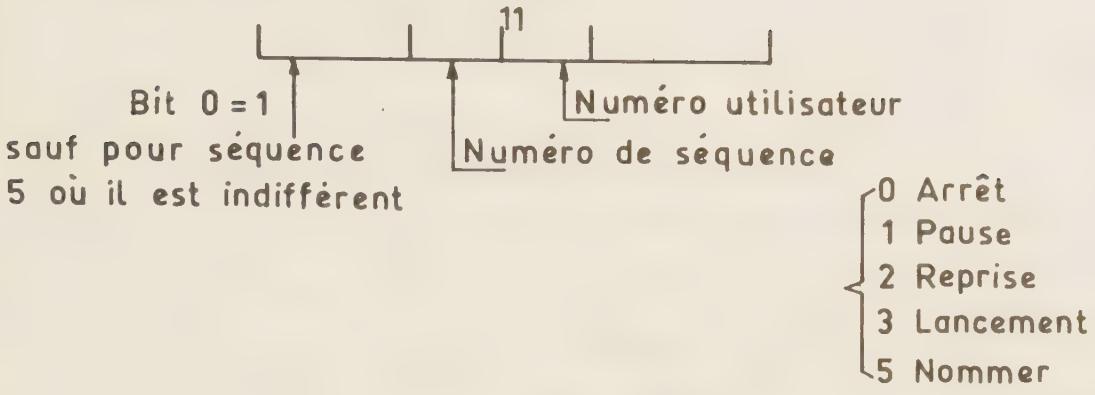
CAV + PTU <PTU à appeler dans A>

IRV AFIN

Dans ce cas, l'argument transmis à la tâche appelée est le numéro utilisateur.

Pour les commandes FORTRAN, il y a choix d'un numéro de séquence (Arrêt, Pause, Lancement, Reprise, Nommer) et appel de la tâche JTLANS.

L'argument transmis est alors :



Pour les commandes spéciales

- LOK fait appel à IMPRI pour mettre la TTY hors tension.
- BAF fait appel à un SP DETACH pour détacher la TTY utilisateur.
- DIS fait appel à la tâche INDIS qui initialise le Display.

IV.7 - Commandes Standards de COSYNUS

Actuellement dix commandes sont à la disposition de l'utilisateur :

- EFP : entrée d'une pile de dix noms maximum de programmes FORTRAN
- RFP : lancement d'un programme FORTRAN
- PFP : suspension " " "
- SFP : Arrêt " " "
- RSP : Reprise " " "
- NFP : Impression du nom du programme en cours
- LOK : Verrouillage du système de commande
- HSC : Modification d'adresse d'un module CAMAC
- DIS : Accès aux divers modes de visualisation sur le T 4002
- BAF : Accès au Background et aux demandes Background.

IV.7.1 - Commande EFP

Cette commande permet donc l'entrée d'une pile de noms de programmes FORTRAN.

Un nom de programme FORTRAN est un ensemble d'au plus 6 caractères alphanumériques.

Mode d'utilisation :

1) Sur l'ordre EFP, le système imprime un * et attend l'entrée du nom. La fin de l'entrée est provoquée par la frappe du caractère RC. Si le nom est correct, et si la pile n'est pas pleine, le système attend l'entrée du nom suivant.
   L'ordre d'entrée des programmes dans la pile détermine l'ordre d'enchaînement des divers programmes.
   Pour terminer l'entrée de la pile, l'utilisateur doit frapper le code EOF suivi de RC.
   Le système met alors la TTY hors tension.

Si, après l'entrée d'un nom de programme, l'utilisateur frappe le caractère RC tout seul, à l'exécution le système s'arrêtera automatiquement avant de lancer le programme. L'utilisateur pourra relancer le déroulement par l'ordre RFP.

2) Structure générale de la tâche

La tâche qui effectue la commande de chargement est une tâche bloquée, commune aux 6 utilisateurs. Cette tâche fait appel à la tâche ECL d'entrée clavier propre à l'utilisateur, laquelle tâche la rappelle lorsque l'entrée clavier est terminée. Il y a alors analyse des caractères entrés et traitement correspondant, c'est-à-dire :

- entrée du nom dans la pile s'il est correct
- ré-entrée clavier si le nom est incorrect
- marquage du nom précédemment entré si code RC
- fin de la tâche si code EOF

3) Description détaillée de la tâche

(voir annexe)

IV.7.2 - Commande RFP

Cette commande permet le lancement d'une pile de programmes FORTRAN ; elle lance le programme pointé par le pointeur de la pile programme.

Utilisation :

Sous le message OK n, il suffit de frapper RFP. Le système répond par RC LF. Cette commande ne sera acceptée que si l'état utilisateur est l'état "arrêt FORTRAN".

Le programme lancé est celui désigné par le pointeur de pile, c'est-à-dire :
- le 1er programme entré, si c'est après l'ordre EFP
- le programme suivant dans la pile, après un ordre SFP.

IV.7.3 - Commande SFP

Cette commande permet l'arrêt du programme en cours. Ce programme est arrêté, et s'il y a un échange en cours sur les périphériques standards, l'échange est tué.

S'il y a un échange sur le CAMAC, cet échange se poursuit (positionnement d'un moteur par exemple) mais, une fois terminé, ne se poursuivra pas.

Le programme arrêté est considéré comme abandonné, et sur un ordre RFP en séquence, on réactive le programme suivant dans la pile.

La commande SFP ne peut intervenir qu'après un ordre RFP ou PFP.

IV.7.4 - Commande PFP

Cette commande effectue une suspension du programme FORTRAN en cours. On termine les E/S qui ont été lancées. Le programme suspendu est prêt à être relancé sur l'ordre RSP. L'ensemble des ordres PFP, suivi de RSP est équivalent à un ineffectif pour le programme.

IV.7.5 - Commande RSP

Cette commande permet la reprise du programme en cours après un ordre de suspension. Elle n'est acceptée qu'après un ordre PFP.

IV.7.6 - Commande NFP

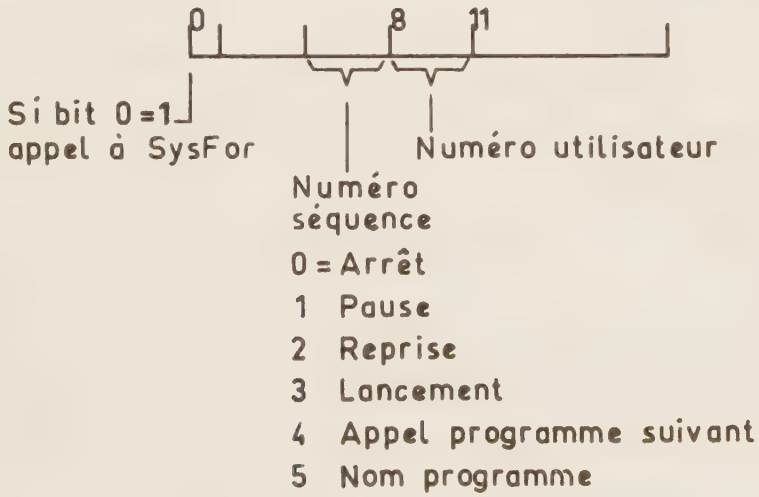
Cette commande provoque l'impression du nom du programme en cours. En particulier, après un ordre SFP, NFP, provoque l'impression du nom du prochain programme à lancer dans la pile.

Les 5 commandes précédentes (RFP, SFP, PFP, RSP, NFP) sont gérées par une même tâche JTLANS qui peut être activée par COSYNUS ou à partir de SYSFOR ou de MONAC.

Cette tâche gère l'ensemble de ces commandes en :

- vérifiant les compatibilités d'état
- remettant à jour l'état utilisateur après chaque commande acceptée
- testant les défauts secteurs généralisés ou par expérience
- gérant la pile des programmes à activer
- faisant les appels à SYSFOR pour activer le FORTRAN proprement dit.

La tâche JTLANS est commune à tous les utilisateurs ; c'est une tâche bloquée. On lui passe en argument.



Elle fait appel à la tâche MODULS de SYSFOR ou à la tâche IMPRI pour imprimer :

- ACHS : si défaut expérience
- OVER : si plus de programme dans la pile
- WAIT : si le programme était "marqué" lors de l'entrée par EFP

IV.7.7 - Commande LOK

Cette commande permet de verrouiller le système, c'est-à-dire que l'Appel Superviseur n'active plus TRCP mais TACLE qui demande le mot de passe. Cette commande n'active pas de tâche spéciale, mais est traitée dans ANALYS.

Sur la commande LOK on range dans la Table des PTU activés par l'Appel Superviseur le PTU de TACLE à la place de celui de TRCP et on fait appel à IMPRI pour

mettre la TTY hors-tension.

IV.7.8 - Commande BAF

Cette commande permet d'affecter le Background à la TTY de l'utilisateur qui en fait la demande.

Dans ANALYS, on regarde dans STBAF si le Background est déjà retenu ou non. Si oui, on refuse ; sinon, on range dans l'US Background l'adresse de la Table UP de l'utilisateur et on change le numéro de l'utilisateur en + 7 qui est le numéro de Background.

Sur la commande "fin de Background", on repasse l'utilisateur sous le contrôle de COSYNUS et on remet STBAF à zéro.

IV.7.9 - Commande HSC

Cette commande permet de changer l'adresse d'un module CAMAC à l'intérieur d'un même chassis, sans changer le système d'IT.

Le système demande :

- le nom de l'USASM attaché au module
- le niveau d'IT
- le sous-niveau
- la nouvelle adresse

Si tout correspond, le changement est fait. Sinon, il y a impression d'un message d'erreur et les tables de MONAC restent inchangées.

La tâche qui réalise la commande, DUCONF, est commune à tous les utilisateurs ; c'est une tâche bloquée.

Elle travaille sur les tables résidentes de MONAC.

Tout d'abord, il y a recherche dans la table de l'utilisateur du nom de l'USASM entré au clavier. S'il figure dans la table, on demande le niveau, le sous-niveau puis il y a recherche dans les tables d'interruption pour vérifier s'il y a l'ancienne adresse. Si oui, on remet à zéro l'ancienne adresse et on charge la nouvelle adresse entrée au clavier. Ensuite, la tâche se termine par un sous-programme qui détache la TTY et remet à zéro TABOU.

IV.7.10 - Commande DIS

Cette commande permet l'accès au dialogue DISPLAY. Cette tâche est unique mais, est accessible par tous les utilisateurs, un seul à la fois.

Le système de dialogue pour le DISPLAY est décrit plus en détail dans un des paragraphes qui suit.

IV.8 - Etats utilisateurs

Dans un système temps réel, il faut faire attention à la compatibilité des commandes, c'est-à-dire aux possibilités d'enchaînement, qui ne doivent pas tous être permis. En effet, certaines commandes ne doivent être suivies que d'un groupe bien défini d'autres commandes sous peine de casser le système. Par contre, certaines commandes, consécutives à une autre, sont dangereuses car elles risquent de modifier des paramètres dans la com-

mande antérieure.

Dans le cas du système CARINE, il faut faire très attention, lorsqu'on a un programme FORTRAN en cours, de ne pas autoriser toutes les commandes, bien que cela soit possible du point de vue système proprement dit, afin de ne pas risquer de modifier par une fausse manoeuvre des tables utilisées par le programme FORTRAN, ce qui pourrait "casser" le système.

Chaque utilisateur a donc été doté d'un "état", qui est significatif du travail qu'il est en train de faire. Le passage d'un état à un autre ne peut se faire que suivant un graphe bien défini, qui exclut toutes les manoeuvres dangereuses. La sécurité de fonctionnement du système s'en trouve donc accrue.

On a défini 6 états utilisateurs :

- état 1 : FORTRAN en cours
- état 2 : VISU + Fortran en cours
- état 3 : Suspension Fortran
- état 4 : Suspension Fortran + VISU
- état 5 : Arrêt FORTRAN ou indifférent
- état 6 : VISU seule

Le graphe de passage d'un état à l'autre est le suivant :

- en dehors de l'arrêt Fortran, toutes les commandes Hors Fortran sont interdites excepté :
  - LOK pour verrouiller,
  - DIS pour activer le Display.
- Si le Fortran est en cours, seules sont autorisées :
  - arrêt,
  - suspension.
- Si le Fortran est suspendu, seules sont autorisées :
  - reprise,
  - arrêt.

Le schéma de la figure 13 résume ces diverses compatibilités.

IV.9 - Système de gestion du DISPLAY

IV.9.1 - Rappels sur le DISPLAY

Il s'agit d'une unité de Visualisation T 4002 de chez Tektronix équipée d'un tube mémoire. Il n'y a donc pas de problème de rafraîchissement de l'image ; par contre, lorsqu'on veut modifier un détail de l'image, il faut tout effacer et régénérer l'image modifiée, ce qui est parfois gênant.

La dimension de l'écran est de 21 x 15,4 cm, ce qui correspond à :

- 39 lignes de 84 caractères alphanumériques
- 1024 points en X et 760 points en Y

La T 4002 peut fonctionner selon 4 modes :

- mode US (alphanumérique)

Ce mode permet l'échange des caractères ASCII comprenant :

. l'alphabet

. les chiffres

. les caractères habituels : point, signe, virgule, etc....

. certains caractères spéciaux permettant de changer de mode.

La taille des caractères peut être simple ou double.

L'écriture italique est également possible.

- mode RS (incrémental)

Ce mode permet de déplacer le spot par rapport à une position donnée d'un incrément dans une direction donnée parmi 8 : Nord, Nord-Ouest, Ouest, Sud-Ouest, Sud, Sud-Est, Est, Nord-Est.

Le spot peut être choisi allumé ou étaint.

- mode FS (point par point)

il positionne le spot en un point défini par ses deux coordonnées X et Y. Le spot est toujours allumé.

- mode GS (vecteur)

permet d'obtenir un vecteur défini par les coordonnées X,Y du point initial et celles du point final.

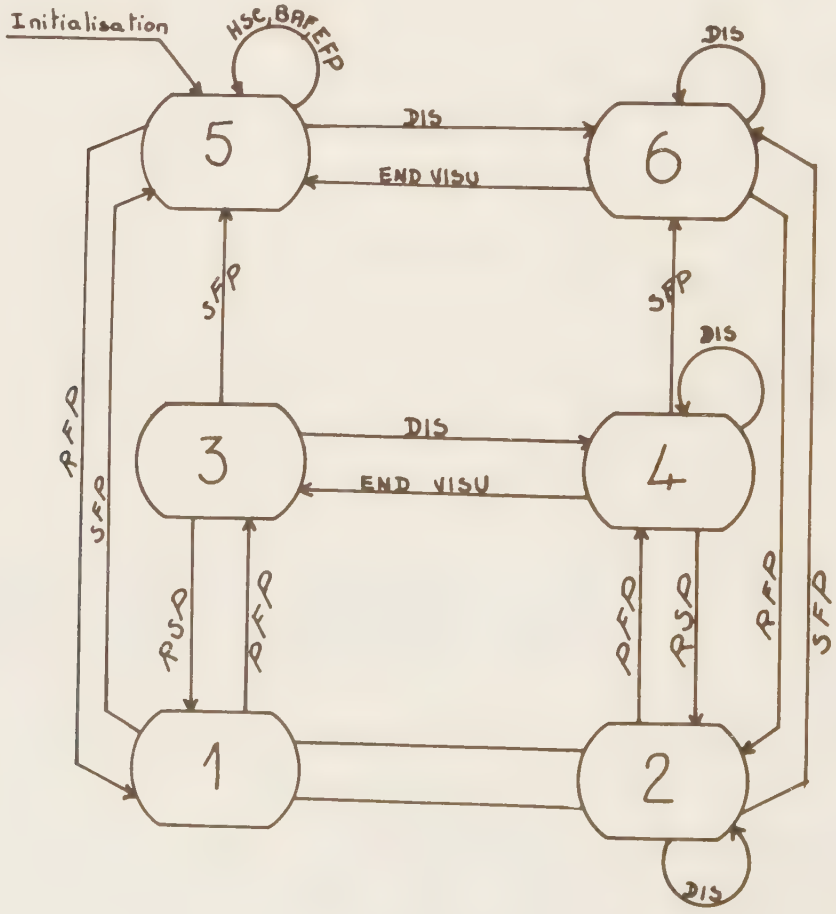


Figure N° 13

Graphe de passage des états " Utilisateur " Cosynus

Au Display se trouve couplé un dispositif permettant d'obtenir une photocopie de l'image apparaissant sur l'écran : c'est le HARDCOPY. Il peut être télécommandé en envoyant au Tektronix un code ASCII spécial appelé ETB.

Une temporisation fait diminuer au bout de 4 minutes l'intensité de l'image, afin de prolonger la vie du tube mémoire. L'envoi de tout caractère au Display redonne à l'image la luminosité normale.

L'unité de Visualisation est couplée au calculateur T 2000 par l'intermédiaire d'un canal automatique non multiplexé. Le Display peut provoquer l'apparition des défauts suivants :

- Hors tension : le display n'est pas alimenté
- LOCAL : le display a été passé en local
- Page Full : la page correspondant à l'écran est pleine
- Hard Copy Hors tension : on a envoyé le code ETB mais le Hard Copy n'est pas alimenté
- Hard Copy défaut papier : on a envoyé le code ETB alors qu'il n'y a plus de papier photosensible dans le Hard Copy.

IV.9.2 - Modes de visualisation

Le physicien a besoin de visualiser les résultats d'une mesure sous forme de courbe. Ces résultats doivent être mis sous forme d'un "bloc de données" comportant une organisation spéciale décrite plus loin. Les courbes doivent être représentées en point par point.

Deux modes sont actuellement disponibles :

- mode 1 : 1 courbe unique
- mode 2 : comparaison de plusieurs courbes (maximum 4)

Un 3ème mode est envisagé donnant une représentation à 3 dimensions dans l'espace.

Dans tous les cas, on doit disposer d'un système de pointage souple, permettant d'obtenir la valeur en X et en Y réelle de chaque point. On doit pouvoir de même faire des extensions (effet de loupe) sur la courbe ainsi que des intégrations de pic.

IV.9.3 - Blocs de données

Les utilisateurs disposent de plusieurs fichiers différents pour le stockage de leurs données au niveau du FORTRAN. Chaque fichier a une taille inférieure à 4K. Chaque utilisateur a un minimum de 4 fichiers appelés par des numéros.

Ces fichiers constituent une zone de stockage temporaire pour un programme FORTRAN et pourront être transférés au choix vers un autre périphérique de sortie (VISU - Bandes magnétiques - Traceurs, etc...). Pour rendre possible le dépouillement et l'exploitation des résultats de mesure, ces derniers sont stockés sous un format bien précis dans les fichiers utilisateurs, et forment ce que l'on appelle un bloc de données.

Un bloc de données aura une structure différente selon qu'il est sur disque ou recopié sur bande magnétique.

Il est composé :

- des données brutes proprement dites
- d'un bloc explicatif des données avec les paramètres de l'expérience HLB 2
- d'un bloc propre à la gestion des bandes magnétiques HLB 1

La configuration sur bande est donc la suivante :

Sur disque seul le bloc HLB 2 demeure, ce qui donne :

La Visu travaillant essentiellement sur des blocs de données présents sur disque, nous n'allons donner une description détaillée que de HLB 2.

Un bloc de données est découpé en cinq parties principales :

- Bloc identification : ce bloc est destiné à identifier l'utilisateur et son expérience, sans tenir compte des acquisitions.

  On y trouve :

- nom de l'expérience	:	4 lettres	2 mots
- N° d'ordre	:	4 chiffres	2 mots
- caractère de contrôle	:	1 chiffre	1 mot
- date de création			3 mots
- heure de création			2 mots
- type de cristal			3 mots
		Soit en tout =	13 mots

- Bloc commentaire : cette zone est destinée à recevoir des paramètres fixes plus une ligne de commentaire. La dimension est donc variable.

  On a donc :

- 1 ligne Télétype 72 caractères	:	36 mots
- nombre de paramètres N_1	:	1 mot
- Icode du paramètre 1	:	1 mot
- valeur du paramètre 1	:	2 mots
'		
'		
'		
N fois	Soit en tout	37 + 3 N1 mots

- Bloc définition du SCAN :

  Ici, on s'intéresse exclusivement au type d'acquisition et aux données et paramètres :

- Type de manipulation :	1 mot
- Nombre de paramètres du SCAN : N_2	1 mot
- Icode de base de SCAN	1 mot
- Valeur de départ	2 mots

- Pas 2

- Nombre de points du scan M 1 mot

Soit en tout $3 + 5\,N_2$

- Bloc de Description du Data Set :

Ce bloc définit l'organisation des données d'acquisition (obtenues à partir du CAMAC) et impose strictement la suite des données dans le Data Set :

- Nombre de paramètres par point d'acquisition : N_3 1 mot
- Icode 1
- Icode 2

etc..... Soit $1 + N_3$ mots

- Data Set proprement dit :

(Valeur Icode 1) N_3 fois
(Valeur Icode 2)

M fois

(Valeur Icode 1
(Valeur Icode 2

Soit : $2 \times M \times N_3$ mots

La longueur totale d'un fichier est donc :

$L = 54 + 3\,N_1 + 5\,N_2 + N_3\,(1 + 2\,M)$ mots

Il suffit donc de connaître les 4 valeurs N_1, N_2, N_3 et M pour connaître parfaitement le bloc de données.

Pour passer de disque à bande, il suffit de générer HLB1 et TLB1 et de recopier HLB2 et le Data set tels quels, à partir du disque.

IV.9.4 - Présentation de l'image

Après une étude qui a permis de concilier les désirs des physiciens et les impératifs de place sur l'écran VISU, on a adopté la représentation suivante :

- En haut de l'écran et sur 3 lignes, écrits en double grandeur, on trouve :
  - le nom de l'expérience
  - le nom du spectre
  - la date de création
  - l'heure de création
  - le nom et la valeur des 6 premiers paramètres figurant dans le bloc identification sous la forme :

    THE = + 123.25
- Ensuite, la ligne de commentaire en grandeur normale
- Le reste de l'écran est occupé par les axes et les courbes représentées en point par point.

On affiche le nom des variables représentées sur les axes. Pour chaque courbe on indique les valeurs de :

- X minimum
- X maximum
- Y maximum
- Y minimum
- X du maximum en Y

Les informations complémentaires (pointages, intégrations) seront affichées à droite de l'écran, en colonne.

IV.9.5 - Organisation du système de gestion de la VISU

Le principe adopté est que l'on affiche sur l'écran de la Visu le bloc ou les blocs de données contenus dans les fichiers FORTRAN indiqués par l'utilisateur avec le cadrage optimal et toutes les informations indiquées plus haut. Cette première image constitue une image de base. Ensuite, par un jeu de commandes dynamiques, l'utilisateur peut faire des pointages, des effets de loupe, des intégrations et des photocopies de l'image.

Il faut signaler aussi qu'il n'y a qu'une unité de visualisation pour les 6 utilisateurs. Donc, il n'y a qu'un seul utilisateur à la fois qui puisse avoir accès à la VISU. L'image est reprise par une caméra de TV et retransmise sur 6 moniteurs de télévision fonctionnant en circuit fermé.

Le principe du tube mémoire rendant impossible l'effacement d'une partie seulement de l'écran, il a été décidé de créer sur disque un fichier qui serait la représentation de l'image en codes VISU, c'est-à-dire qu'il suffit, pour régénérer l'image, de faire un simple échange entre le disque et le DISPLAY. Ainsi, pour effacer une partie de l'image, on efface sur disque la partie concernée, et on régénère sur la VISU l'image ainsi modifiée.

Deux tâches principales de classe 2 contrôlent la gestion des programmes de visualisation :

- INDIS : qui dialogue avec l'utilisateur afin d'entrer les données nécessaires à la génération de l'image de base sur la VISU
- SUPDIS : qui assure l'entrée des commandes dynamiques, permettant les diverses opérations sur l'image de base.

Ces deux tâches utilisent d'autres tâches de classe 2 ayant des fonctions bien définies (affichages des commentaires, tracé des axes, etc...) que nous verrons plus loin.

IV.9.6 - Commandes VISU

Nous avons vu que c'était la commande DIS qui permettait l'accès au système de gestion de la VISU. Cette commande active la tâche INDIS qui demande :

- le mode envisagé
- le numéro du fichier FORTRAN à VISUALISER
- les unités en X
- les unités en Y

On obtient alors, sur l'écran, l'image de base. Le contrôle est ensuite donné

à la tâche SUPDIS qui reconnaît les commandes dynamiques, qui sont les suivantes :

- AXE n

numéro de fichier

Cette commande permet de générer sur l'écran un ou deux axes de pointage (selon le mode choisi), sur le premier point de la courbe.

On peut alors entrer les sous-commandes :

- H : déplacement vers le haut
- B : déplacement vers le bas
- G : déplacement vers la gauche
- D : déplacement vers la droite

Ces axes sont générés par un dispositif Hardware, qui fait en sorte qu'ils sont visibles, mais non mémorisés. Si on entre la sous-commande "M", l'axe est mémorisé et on affiche sur l'écran les valeurs X et Y correspondantes.

Pour obtenir à nouveau l'image initiale, sans les axes mémorisés et les valeurs correspondantes, on frappe la sous-commande "Z" qui régénère une image propre.

Pour ramener les axes sur le premier point de la courbe, il suffit de taper la commande "C".

Après l'entrée et l'exécution de chaque sous-commande, l'utilisateur est en attente de l'entrée au clavier de la sous-commande suivante. Pour en sortir, il faut frapper la sous-commande "Q", qui ramène sous SUPDIS, prêt à entrer une commande.

- EXT : Cette commande permet d'avoir un effet de loupe sur la ou les courbes entre deux valeurs X_1 et X_2 entrées par l'utilisateur.
- INT : Cette commande permet d'obtenir l'intégration de la ou les courbes entre deux valeurs X_1 et X_2 entrées au clavier.
- ETB : Cette commande est utilisée pour déclencher le système HARD COPY.
- DIS : grâce à cette commande, on repasse sous le contrôle d'INDIS qui permet de travailler sur un autre fichier ou de changer de mode.
- END : Cette commande libère la VISU et la rend disponible pour un autre utilisateur.

IV.9.7 - Tâches utilisées par la VISU

Outre les deux tâches INDIS et SUPDIS, dont nous avons déjà parlé, et qui sont les deux principales du point de vue organisation, le système de gestion de la VISU utilise les tâches suivantes :

- AFFCOM : Cette tâche va chercher la zone commentaire du bloc de données et la réarrange pour l'afficher sur la VISU.
- AFFAXE : Cette tâche trace les axes repères
- MINMAX : Cette tâche effectue la recherche des maxima et minima en X et en Y afin d'obtenir le cadrage optimal.
- COURBE : Cette tâche génère l'image point par point de la courbe sur l'écran.
- AFFVAL : Affiche les valeurs remarquables des courbes sur l'écran.
- POINTE : Gère tout le dispositif de pointage sur l'écran.
- EXTENS : effectue l'effet de loupe sur une partie de la courbe.

- INTEGR : Calcule l'intégrale de la courbe entre deux points
- ENDVIS : Effectue toutes les réinitialisations et autres travaux nécessaires à la fin de l'utilisation de la VISU.
- AFFDIS : Effectue les transferts entre l'image sur disque et la VISU proprement dite.

IV.9.8 - Tâche INDIS

Comme nous l'avons déjà dit, cette tâche assure le dialogue nécessaire à l'obtention d'une première image sur l'écran.

Elle entre le mode, puis demande le ou les numéros des fichiers FORTRAN à visualiser (4 au maximum).

Nous avons vu que quatre paramètres (N_1,N_2,N_3,M) suffisaient à définir un bloc de données. Pour chaque numéro de fichier entré, INDIS va donc chercher sur disque les 4 valeurs caractéristiques par l'algorithme suivant :

- lire le 14ème mot du fichier = N_1
- progresser de $3 * N_1 + 37$ dans le fichier
- lire le mot = N_2
- progresser de $5 * N_2$
- lire les deux mots qui suivent = M et N_3

Ces valeurs sont rangées dans 4 tables résidentes et seront indispensables pour la suite. On range également en résident le nombre de fichiers à visualiser (≤ 4).

INDIS entre ensuite les noms à afficher sur les axes, représentant les noms de variables.

Ensuite, INDIS fait appel à :

- AFFCOM pour les commentaires
- AFFAXE pour les axes
- AFFVAL)
- COURBE (pour les courbes
- MINMAX)
- AFFDIS pour générer l'image sur la VISU

puis donne le contrôle à SUPDIS pour entrer les commandes dynamiques.

IV.9.9 - Tâche AFFCOM

Cette tâche transforme donc la zone commentaire du bloc de données pour la rendre compatible avec le display.

La particularité de cette tâche est qu'elle travaille sur un fichier disque pour la correspondance Icode → NOM

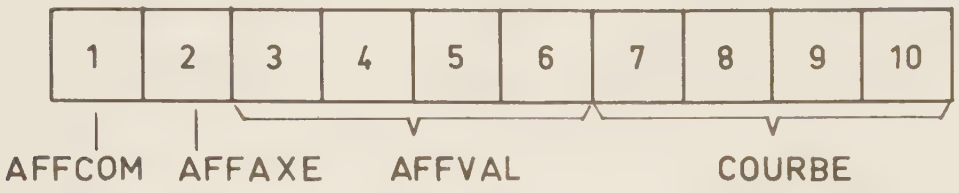
En effet, on a vu que les variables (Théta, Oméga, Phi, etc...) étaient symbolisées en FORTRAN par Icode, donc un nombre et qui figure donc dans le bloc de données. Afficher un nombre sur le display pour une variable serait peu parlant. Aussi a-t-on sur disque une table de correspondance entre les Icodes de chaque utilisateur et le nom (3 caractères) de la variable. AFFCOM effectue la recherche nécessaire sur disque pour retrouver les noms correspondants.

IV.9.10 - Tâches AFFAXE, AFFVAL, COURBE, MINMAX

Ces tâches effectuent des calculs trop complexes et longs pour être décrits ici. Mentionnons cependant quelques-unes de leurs particularités.

Ces tâches, de même que AFFCOM, travaillent avec le disque, c'est-à-dire qu'elles ne génèrent pas d'image sur la VISU, mais sur une zone disque représentant en codes VISU l'image qui apparaîtrait sur l'écran.

On utilise pour cela un fichier de SYSFIC, baptisé DIS, et qui comporte 10 articles comme le montre le schéma suivant :



On remarque que AFFVAL et COURBE ont 4 articles car on peut avoir la comparaison de plusieurs courbes, jusqu'à un maximum de quatre.

Les tâches AFFVAL, COURBE, MINMAX ont été écrites de façon générale, afin de pouvoir les utiliser pour faire des effets de loupe. Lorsqu'on les active, on leur passe en argument :

- le nombre de points à sauter par rapport au début du "data set"
- le nombre de points à traiter à partir du point indiqué ci-dessus
- le numéro de la courbe (1 à 4)

On peut ainsi utiliser ces tâches à partir de toutes les autres pour obtenir les effets désirés quant à la représentation sur l'écran.

IV.9.11 - Tâches EXTENS et INTEGR

Ces tâches demandent l'entrée au clavier des deux valeurs limites de X, X_1 et X_2, définissant la portion de la courbe concernée.

EXTENS utilise AFFVAL et COURBE pour générer l'image grossie de la portion de courbe.

INTEGR calcule l'intégrale selon la méthode classique des trapèzes et l'affiche directement sur la VISU.

IV.9.12 - Tâche AFFDIS

Cette tâche génère une image sur l'écran du DISPLAY à partir de la zone disque représentative. Le transfert a lieu par blocs de 80 mots.

On a utilisé pour ces transferts une particularité de SYSFIC, complétée par la création d'une nouvelle commande pour SYSFIC.

En effet, SYSFIC possède dans sa table des articles, la longueur actuelle de ces articles. Or :

- en lecture, si l'on demande à lire plus loin que la longueur actuelle, SYSFIC ne lit que la longueur actuelle et fait un compte-rendu d'erreur.

- en écriture, la longueur actuelle n'est remise à jour que si on déborde de la valeur actuelle.

Cette particularité est très intéressante pour AFFDIS, car on ne connaît pas à priori la longueur de certains articles de la zone disque VISU, et c'est plus commode que d'avoir des tables de longueur d'articles, remises à jour à chaque opération. Malheureusement, la longueur actuelle de l'article n'est pas forcément remise à jour.

On a donc transformé la commande "créer" de SYSFIC, qui permettait de créer dynamiquement des articles à l'intérieur d'un fichier, et qui était inutilisée dans le système CARINE, en une commande qui effectue la remise à zéro de la longueur actuelle de l'article. Ainsi, à chaque écriture, la longueur actuelle sera nécessairement remise à jour et contiendra la longueur effective de l'article.

Il suffit donc de demander des lectures sur les articles jusqu'à ce que l'on détecte l'erreur "débordement de longueur actuelle". On passe alors à l'article suivant.

IV.9.13 - Tâche SUPDIS

Cette tâche entre les commandes dynamiques en faisant appel à la tâche ECL, qui effectue l'entrée proprement dite. SUPDIS analyse ensuite la commande entrée et effectue le branchement vers les séquences correspondantes :

- pour ETB, on envoie au display le code "ETB" qui active le HARD COPY,
- pour AXE, on active la tâche POINTE, qui gère les pointages,
- pour END, on active la tâche ENDVIS avec un argument négatif,
- pour INT, on appelle la tâche EXTENS avec un argument négatif,
- pour EXT, on appelle la tâche EXTENS avec un argument positif,
- pour DIS, on appelle la tâche INDIS.

IV.9.14 - Tâche ENDVIS

Lorsque la commande DIS est frappée sous COSYNUS (et non sous SUPDIS), on arme une temporisation de 4', afin de ne pas permettre à un utilisateur de monopoliser la VISU.

On dispose de deux mémoires :

- STVISU : qui contient le numéro de l'utilisateur occupant la VISU
- RETVIS : qui contient le numéro du premier utilisateur ayant demandé la VISU alors qu'elle avait été retenue.

Sur la fin de la temporisation (argument = 0) ou sur la commande END sous SUPDIS (argument : 4000), on active la tâche ENDVIS qui effectue les opérations suivantes :

- signaler à l'utilisateur de STVISU qu'on lui supprime l'accès au DISPLAY,
- signaler à l'utilisateur de RETVIS que la VISU est libre,
- effectuer les remises à jour nécessaires (état utilisateurs, flags de protection, etc...).

Toutefois, si à la fin de la temporisation, la mémoire RETVIS est nulle, on prolonge l'utilisateur actuel de la VISU d'une durée de 4' et ce, jusqu'à ce que RETVIS soit différent de zéro.

IV.10 - Sécurité de fonctionnement

La conception du système CARINE a été guidée par un souci de sécurité de fonctionnement. Le réacteur nucléaire fonctionnant par périodes, il est indispensable que toutes les précautions soient prises pour que le système soit fiable, car alors la moindre panne pourrait faire perdre des journées et même des semaines de mesure pour un physicien.

Du point de vue Hardware, le calculateur et les périphériques classiques ont été alimentés sur un dispositif secouru en cas de panne de courant. Les baies CAMAC ne sont pas secourues, mais signalent par un défaut l'absence de courant.

Le software a de même été conçu pour qu'en cas de mauvais fonctionnement ou de fausse manoeuvre de l'utilisateur, tout d'abord il n'y ait que l'utilisateur en défaut qui soit pénalisé, ensuite que ce dernier puisse redémarrer rapidement et avec le minimum de manoeuvres.

Les fausses manoeuvres utilisateurs sont en principe toutes piégées et aboutissent à des erreurs sans gravité, ou dans les cas extrêmes, à l'arrêt du programme FORTRAN de l'utilisateur. Ainsi, sur une panne du système CAMAC, le programme est généralement suspendu, avec impression de la cause de la panne. Le physicien peut alors décider s'il reprend son programme (RSP) ou s'il l'arrête (SFP).

Un point assez délicat a été aussi solutionné. Si, au cours d'un échange système avec le télétype de l'utilisateur, ce dernier la mettait en local, bien sûr l'échange était tué et tout se passait bien au niveau système, mais le Hardware de la Télétype était toujours en défaut (local permanent) et, même lorsqu'on repassait "en ligne", l'Appel Superviseur ne pouvait monter puisqu'il y avait toujours le défaut "local" mémorisé et non remis à zéro. On a donc fait un traitement spécial où l'on envoie à la Télétype un ordre de Mise Hors Tension suivi d'un ordre de mise sous tension qui réinitialise le Hardware.

Un autre problème à résoudre a été celui du redémarrage après un arrêt secteur, dans le cas où le secouru est aussi en panne.

Pour solutionner le problème, on traite l'Alarme Défaut Secteur qui apparaît lors de la disparition du secteur, et à la reprise, le pointeur est initialisé par Hardware à l'adresse 0010 du bloc 0, qui branche sur une séquence particulière de RTDMS, où l'on active la tache zéro (PTU = 0) qui fait la réinitialisation de toutes les tables de RTDMS et de MONES, en mémoire et sur disque, et réinitialise également les variables utilisées par COSYNUS. Le système est alors prêt à fonctionner à nouveau.

Tout cet ensemble de précautions fait du système gérant CARINE un système particulièrement sûr du point de vue fonctionnement.

V - DESCRIPTION DE L'ENSEMBLE SOFTWARE CHARGE DE LA GESTION DU FORTRAN : "SYSFOR"

Préliminaires

Dans ce chapitre, nous allons décrire la partie réalisée par Monsieur VINIT et Monsieur LESOURNE, en collaboration avec Messieurs LUMINEAU et OUDJAOUDI de la Société TELEMECANIQUE.

Comme nous l'avons déjà dit, nous décrivons ce sous-ensemble du système CARINE afin que le présent rapport constitue un tout, bien que nous n'ayons pas personnellement participé à la réalisation, sinon pour les interfaces avec le système COSYNUS.

V.1 - Généralités sur le FORTRAN T 2000

Nous avons vu au tout début que, dans la bibliothèque software du T 2000, figurait une Arithmétique Flottante, de type interprétatif, avec représentation des nombres flottants sur 2 mots de 20 bits. On peut, grâce à cette représentation, travailler avec des nombres de l'ordre de $10^{\pm 80}$. L'interpréteur flottant est réentrant, grâce à une zone de travail de 150 mots définis chez l'utilisateur.

Le FORTRAN T 2000 utilise cette arithmétique flottante, avec quelques modifications minimes, pour sa partie calcul. Il est également de type interprétatif, ce qui est intéressant pour la place mémoire nécessaire, mais coûteux en temps machine nécessaire. On considère qu'en moyenne une instruction FORTRAN est traduite par 10 mots machine de code généré.

La première version du Fortran était une version autonome, fonctionnant en temps différé. Pour les besoins du système CARINE, une version "temps réel" de ce FORTRAN était nécessaire, et les modifications nécessaires pour les liaisons avec RTDMS et le système CAMAC ont donc dû être faites.

Vue d'ensemble du FORTRAN temps réel

Dans le cadre du système CARINE, le FORTRAN doit permettre le déroulement en Time-Sharing de plusieurs programmes FORTRAN, ce déroulement étant contrôlé par les commandes en provenance du système COSYNUS, et étant d'autre part soumis aux impératifs de fonctionnement du système gérant le CAMAC lors d'un appel à un programme spécifique de la bibliothèque CAMAC.

La place mémoire nécessaire à un programme FORTRAN assez gros étant importante, la solution la moins coûteuse en place est d'avoir les n programmes utilisateurs sur disque et de les amener chacun à leur tour en mémoire centrale pour leur exécution. Chaque utilisateur peut ainsi disposer d'environ 8 K de code généré, ce qui permet l'écriture de programmes FORTRAN assez conséquents. Rappelons que les temps libres laissés par les utilisateurs FORTRAN sont utilisés par le Background, qui est un programme de dernière priorité placé au-dessous de la barre de verrouillage.

La mise en place du Compilateur FORTRAN sous RTDMS avec accès simultané par tous les utilisateurs posant de gros problèmes (de place, en particulier), il a été

décidé de mettre ce compilateur en Background et de lui adjoindre une gestion de bibliothèque de programmes donnant plus de souplesse à l'utilisateur.

Sous le background on peut donc :

- compiler du FORTRAN,
- travailler sur la bibliothèque de programmes,
- exécuter du FORTRAN,
- avoir accès à l'assembleur ASMAT T 2000

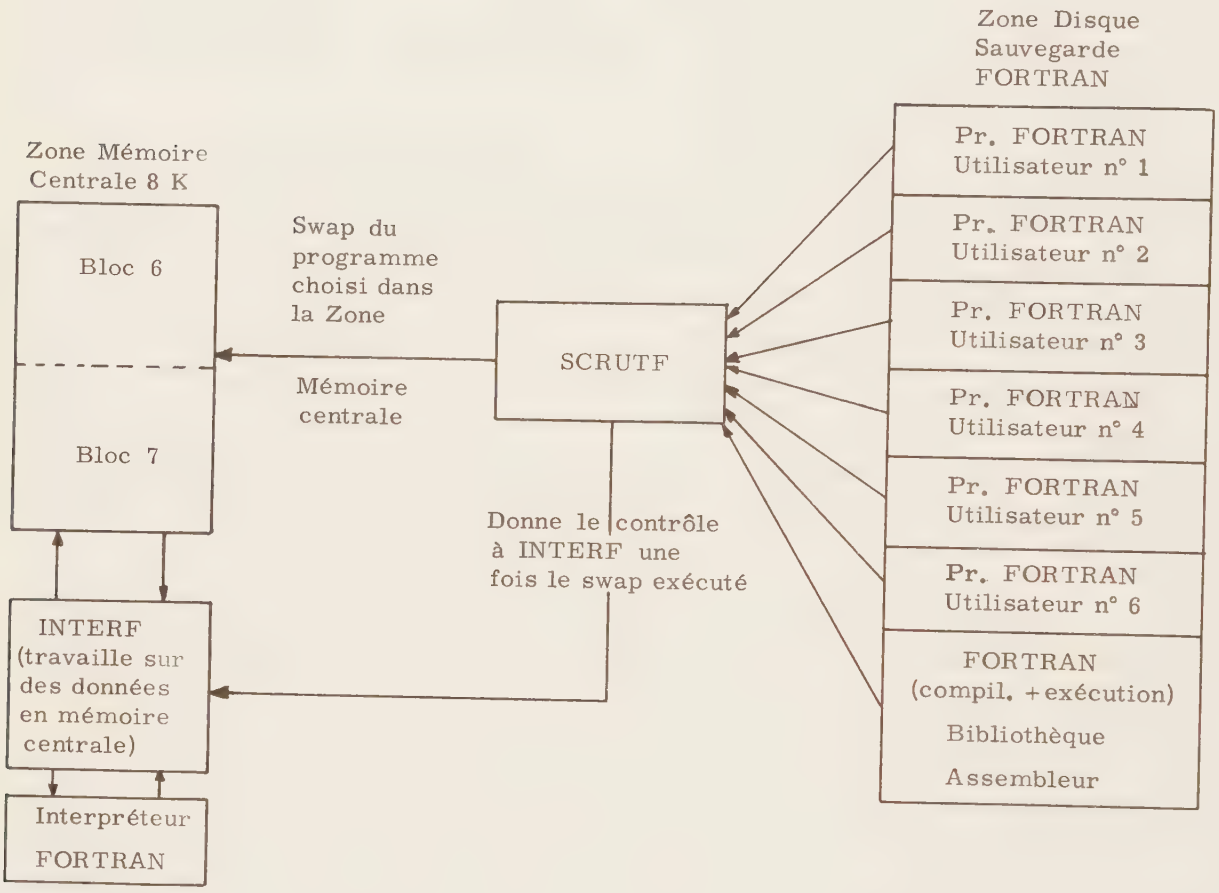


Figure N° 14

Schéma de fonctionnement avec 6 utilisateurs et le Background

Un programme FORTRAN utilisateur est une tâche de classe 2 et fonctionne donc en FOREGROUND : c'est une tâche présente au sens RTDMS. La tâche SCRUTF du système SYSFOR gère la répartition et le swap des 6 utilisateurs ainsi que du background et donne ensuite le contrôle soit à la tâche INTERF, qui fait appel à l'interpréteur Fortran, soit à la tâche BACKGROUND. Pour RTDMS, il n'y a donc toujours qu'un même utilisateur.

Les entrées/sorties sur les périphériques classiques demandées à partir du FORTRAN passent par l'intermédiaire FORES, qui transforme les demandes d'E/S du FORTRAN en tables d'échanges compatibles avec MONES et fait appel ensuite à des tâches particularisées (MONE 1 à 7) qui effectuent les E/S proprement dites.

La figure 14 donne le schéma de principe du fonctionnement avec 6 utilisateurs et un Background.

V.2 - Description de SYSFOR

Le système SYSFOR est donc un ensemble de tâches gérées par RTDMS permettant le déroulement de plusieurs programmes FORTRAN, dont un Background, en simultanéité pour les utilisateurs.

Ce système traite un certain nombre d'évènements externes ou internes au programme FORTRAN en cours ou susceptible d'être activé :

- Commandes utilisateurs (transmises par COSYNUS)
  - . lancement de programme
  - . suspension
  - . arrêt
  - . reprise
- Interruption de Time-Sharing en provenance du cadenceur, et qui limite à 4 secondes le temps maximum de résidence en mémoire centrale d'un programme utilisateur à chaque cycle.
- Défauts en provenance du Système CAMAC
- Demande et fin d'échanges sur périphériques classiques
- Demande et fin d'échanges sur le système CAMAC
- Appels Superviseurs Télétype Background

Les différentes tâches de SYSFOR, ainsi que leurs liaisons, sont représentées dans le schéma de la figure N° 15.

Nous allons donner le rôle de chacune de ces tâches.

V.2.1 - Tâche GESBAC

Cette tâche assure la gestion du Background, c'est-à-dire :

- Activation
- Désactivation
- Réinitialisation

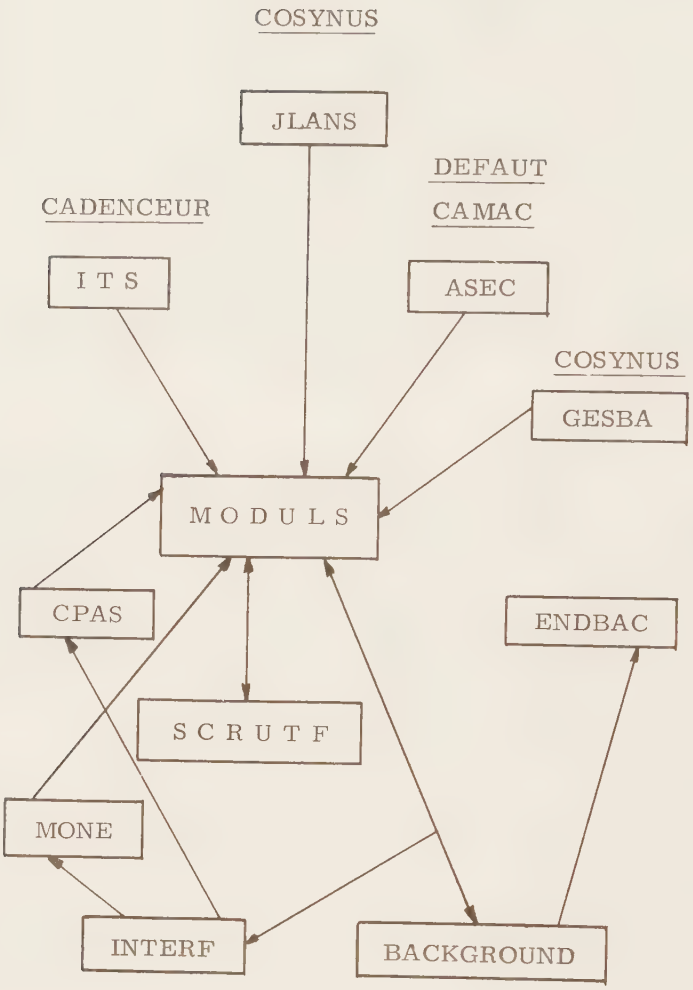


Figure N° 15

Vue d'ensemble de SYSFOR

Activation

La commande "BAF", reconnue par la tâche ANALYS du système COSYNUS, provoque l'appel de GESBAC avec un argument positif. De même, l'appel superviseur de la TTY Background provoque l'appel de GESBAC, mais avec un argument nul, ce qui permet un aiguillage.

On positionne à 1, deux flags, ETABAC et INIBAC et on appelle MODULS, qui activera le Background au moment voulu.

Désactivation

Sur la clé ENB (End of Background), le superviseur Background BOSF (voir plus loin) appelle GESBAC qui remet à zéro l'indicateur ETABAC et fait les remises à jour dans les tables d'US de la TTY background.

Réinitialisation

L'appel superviseur de la TTY auquel est affecté le Background provoque la mise à 1 de INIBAC et l'appel de MODULS. Le positionnement de INIBAC à 1 provoque la réinitialisation du Background par SCRUTF.

Il existe également un indicateur appelé FLASQ, qui s'il est positionné, interdit la réinitialisation, car il signifie que le Background est en train de retasser la bibliothèque.

V.2.2 - Tâche ITS

Cette tâche, activée par l'interruption du cadenceur, interrompt le déroulement d'un programme FORTRAN d'un utilisateur et provoque le passage au programme suivant activable.

L'appel en provenance du cadenceur a une action effective si les trois conditions successives suivantes sont réalisées :

- il doit y avoir plus d'un utilisateur actif
- la tâche SCRUTF doit être hors déroulement
- le programme à interrompre était effectivement actif

On désactive alors la tâche INTERF, et on appelle MODULS qui fera le changement d'utilisateur.

V.2.3 - Tâche MONE

Cette tâche est appelée séquentiellement par INTERF lors d'une demande d'E/S classique, avec pour argument l'adresse d'une table d'échange résidente. Mais, sur sa fin, MONE ne doit pas réactiver INTERF, comme le voudrait la logique de RTDMS, mais on doit faire appel à MODULS qui est chargée de tous les problèmes d'activation, réactivation, mise en attente des programmes FORTRAN. On modifie les tables de RTDMS pour éviter ce rappel.

- s'il s'agit d'une E/S autre que le disque, on appelle MODULS qui met en attente longue l'utilisateur demandant l'E/S
  - . on fait appel à FORES pour exécuter l'échange
  - . on fait appel à MODULS pour la fin d'attente longue lorsque l'échange est terminé.
- s'il s'agit d'une E/S avec le disque :

. on fait un appel séquentiel à FORES

. on appelle MODULS qui réactivera INTERF sous certaines conditions.

V.2.4 - Indicateurs de SYSFOR

Pour la bonne compréhension de la suite, nous allons donner le nom et le rôle des principaux indicateurs utilisés par SYSFOR :

- ETAF : 1 mot par utilisateur, indiquant l'état du programme FORTRAN. Seuls les 3 bits de gauche sont significatifs :
  . 000 : Nul
  . 001 : Inactif Interrompu
  . 010 : Inactif - Attente longue
  . 100 : En cours
  . 101 : Actif Interrompu
  . 110 : Actif attente longue

  La signification des états est la suivante :
  - Nul : n'a pas été lancé ou a été arrêté
  - Inactif : le déroulement du programme a été suspendu
  - Attente longue : le programme est en E/S avec un périphérique classique ou travaille avec le CAMAC
  - Interrompu : le programme a été interrompu par le cadenceur
  - Actif : le programme a été lancé, et est ni suspendu, ni arrêté
  - En cours : le programme est en cours de déroulement

  Le bit 12 de ETAF indique si le programme est présent (= 1) ou non (= 0) en mémoire centrale.

Le passage d'un état à l'autre est géré par le graphe représenté sur la figure N° 16

- NBPFAC : 1 mot. Contient le nombre de programmes activables (état "en cours" ou "actif interrompu")
- IMONE : 1 mot par utilisateur contenant
  - 0 si pas d'E/S
  - le numéro de l'US si échange demandé avec MONES
- PADJU : 1 mot
  - bit 0 : représente défaut général
  - bit 1 à 6 : 1 bit défaut par expérience
- ADCHG : 2 mots par utilisateur. Contiennent le nom du programme que l'on vient d'appeler
- HORS : 1 mot contenant une adresse relais vers une mémoire où sont enregistrés les tops (1s) du cadenceur
- PTFC : 1 mot contenant le numéro de l'utilisateur dont le programme est en train de se dérouler. A l'initialisation PTFC = 0
- ALORSE : 1 mot par utilisateur. Contient les états utilisateurs relatifs aux défauts d'expérience :

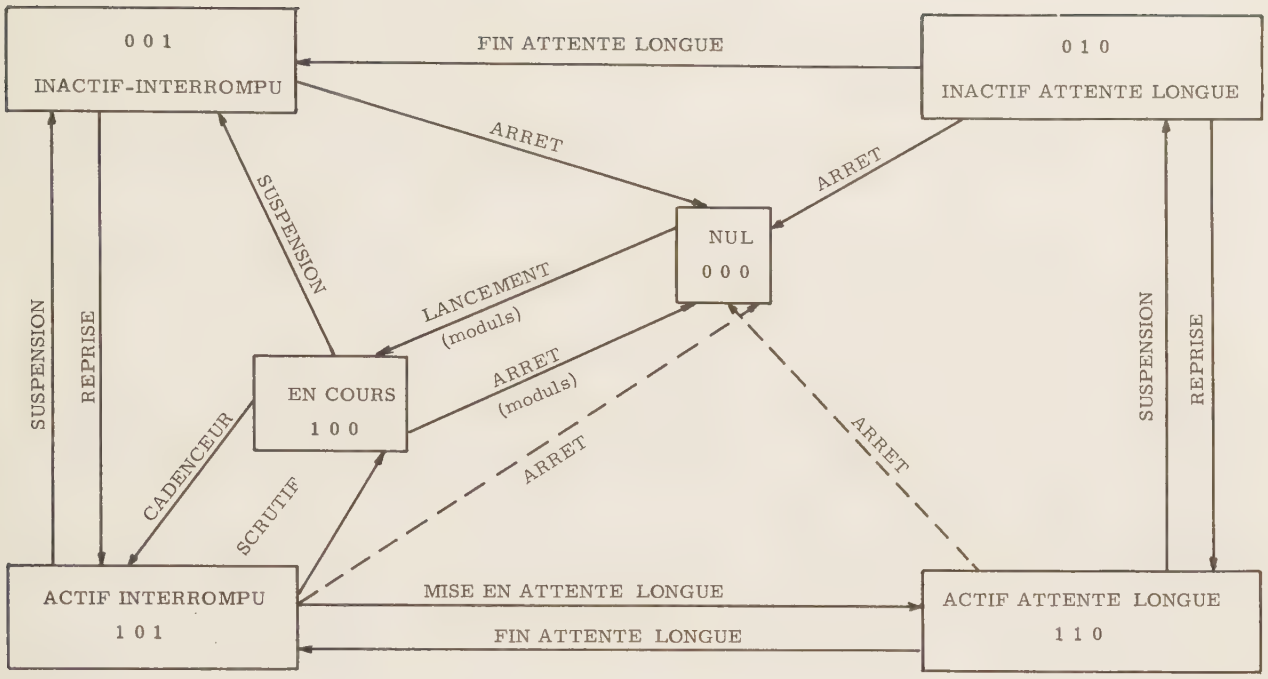


Figure N° 16

Graphe des états FORTRAN

- 0 : Rien à signaler
- 1 : Arrêt - Suspension Défaut
- 2 : Reprise après suspension
- 3 : Lancement après un arrêt
- 4 : Reprise après un défaut

- DEFECH : 1 mot. 1 bit par utilisateur indiquant qu'une demande à FORES a été refusée.
- BACOK : background présent
- ETABAC : background actif
- INIBAC : background à initialiser

V.2.5 - Tâche ASEC

Cette tâche est activée à la suite d'un défaut. Un argument 0 indique une panne générale, sinon c'est le numéro de l'utilisateur en défaut.

- si le programme est actif ou en cours, la tâche ASEC fait appel à MODULS pour suspendre le programme et positionne dans PADJU un bit qui indique à COSYNUS qu'il y a eu suspension et que seules sont permises les commandes "Arrêt" ou "Reprise"
- si le programme n'est pas dans l'état nul, la tâche positionne seulement un bit dans PADJU, sinon il n'y a aucune action

- dans le cas d'une panne générale, ASEC fait un seul appel à MODULS pour une suspension générale et positionne les bits de PADJU comme précédemment.

V.2.6 - Tâche SCRUTF

Cette tâche assure la gestion des 8K de zone FORTRAN en mémoire. A la suite d'un évènement, SCRUTF sort sur disque le programme occupant pour entrer en mémoire un autre programme activable ou le background.

SCRUTF est toujours appelée séquentiellement par MODULS pour donner le contrôle à l'utilisateur suivant ou au background.

SCRUTF teste si au moins un des programmes est activable (NBPFAC $\neq$ 0) ou si le background est lancé, sinon elle se termine avec 0 dans PTAAP. Si l'un des programmes est activable, SCRUTF effectue la recherche en anneau du premier programme activable à partir du dernier qui a été lancé et qui est mémorisé dans PTFC.

Si aucun programme n'est présent en mémoire, SCRUTF commence par la phase "entrée du programme". Sinon, SCRUTF range sur disque le programme présent. On essaye de faire un swap minimum en testant si le programme FORTRAN est à cheval sur la limite des deux blocs de 4K, sinon on récupère les adresses des zones utiles dans les opérandes FORTRAN, et on effectue deux échanges inférieurs à 4K. S'il s'agit du background, on sort systématiquement les deux blocs entiers. Les deux échanges sont nécessités par le fait qu'on ne peut transmettre plus de 4K mots par le canal automatique en un seul échange.

Avant d'effectuer les échanges avec le disque, on met le programme à sortir dans l'état absent, puis on lance deux appels à MONES, le premier non séquentiel et le second séquentiel non swappable.

Ensuite, on effectue la phase "entrée du programme" :

- si le programme n'a jamais été activé, on fait monter en mémoire un programme initialisé, sinon on remplit les tables d'échange à l'aide d'informations mémorisées au moment de la dernière sauvegarde sur disque du programme.
- on effectue également deux appels à MONES. Une fois le programme en mémoire, on repositionne, s'il y a lieu, la barre de verrouillage et on met le programme à l'état présent et en cours (seulement "présent" pour le Background). Ensuite, on range dans PTAAP le PTU de la tâche à activer qui est, soit INTERF, soit le Background, l'activation de ces tâches devant être faite par MODULS.

V.2.7 - Tâche MODULS

Toutes les tâches de SYSFOR font appel à MODULS pour modifier l'état d'un programme FORTRAN ou du Background, et éventuellement appeler SCRUTF qui effectuera le chargement de programme en Zone Fortran.

L'argument d'appel pour MODULS est :

- 7700 en provenance d'ITS
- 7000 en provenance de GESBAC

Les appels en provenance des autres tâches sont constitués de la façon suivante:

- cadré en 11 le numéro d'utilisateur
  - . 0 général
  - . 1 à 6 utilisateur
  - . 7 background
- cadré en 8 le numéro de séquence
  - . 0 : Arrêt
  - . 1 : Suspension
  - . 3 : Lancement ou reprise
  - . 4 : Fin de programme FORTRAN
  - . 5 : Mise en attente longue
  - . 6 : Réactivation après transfert disque

Les appels parvenant à MODULS peuvent donc provenir de différentes sources :

- Appel provenant de ITS. Sur cet appel on teste si le dernier programme qui s'est déroulé est dans l'état en cours et présent ; sinon l'appel est ineffectif. On passe alors le programme à l'état actif interrompu et on fait appel séquentiel à SCRUTF.
- Appel provenant de Gesbac
  Cet appel provoque directement un appel à SCRUTF
- Demande d'arrêt de programme
  Suivant l'état du programme, il y a diverses actions possibles ; toutes conduisent à mettre le programme à l'état nul et à appeler éventuellement SCRUTF
- Demande de suspension de programme
  D'une manière générale, le programme passe d'un état actif au même inactif avec éventuellement appel de la tâche SCRUTF
- Lancement ou reprise de programme
  C'est un lancement si le programme est à l'état nul : on le met alors à l'état en cours. Sinon, c'est une reprise : on passe alors le programme d'un état inactif au même état actif, éventuellement il y a appel à SCRUTF
- En attente longue
  MODULS fait passer le programme de l'état actif ou inactif attente longue à l'état actif ou inactif interrompu, avec toujours appel éventuel de SCRUTF
- Fin de programme FORTRAN
  Le programme qui était en cours, passe à l'état nul et on appelle la tâche JTLANS qui activera le prochain programme rangé en pile par appel à MODULS.
- Mise en attente longue
  Le programme est mis en Attente longue, et on fait appel à SCRUTF
- Réactivation après transfert disque
  Lors d'une demande d'E/S sur fichier ou sur US disque, la tâche FORTRAN est mise en attente pendant tout le transfert mais le programme FORTRAN n'est pas mis à l'état Attente longue. MODULS rappelle la tâche INTERF en fin de transfert.

V.2.8 - Tâche INTERF

Lorsqu'on lance un programme FORTRAN, la tâche INTERF se branche sur l'interpréteur FORTRAN. Elle comporte d'autre part quatre séquences particulières qui traitent :

- le PAUSE Fortran
- le STOP "
- les demandes d'E/S
- les demandes au CAMAC

C'est une tâche de 8 K présente au sens RTDMS, la tâche SCRUTF gérant l'occupation mémoire des programmes FORTRAN.

A l'appel d'un programme, la tâche INTERF est constituée de deux zones :

- dans le premier bloc une partie programme (100_8 mots) et dans le deuxième bloc une partie zone opérande FORTRAN initialisée (700_8 mots).

Les séquences particulières réalisent les actions suivantes :

- PAUSE : on fait appel à la tâche ASEC qui fera appel à MODULS pour suspendre le programme en cours. On régénère ensuite ALORSE
- STOP : on fait appel directement à MODULS en lui demandant une séquence "fin de programme Fortran" et on termine la tâche par un RPT
- Demande d'E/S : on fait un appel séquentiel à la tâche MONE correspondant au numéro d'utilisateur en cours avec pour argument l'adresse de la table d'échange à FORES. Au retour, on teste si la demande est refusée, auquel cas, on revient en tampon sinon en tampon + 1 (compatibilité avec le FORTRAN)
- Demande au CAMAC : on fait appel au CPAS correspondant au programme en cours avec pour argument l'adresse de la table des paramètres.

V.2.9 - Tâche BACKGROUND

Le Background est constitué par une zone de 8K sous la barre de verrouillage. Il y a un superviseur background BOSF qui occupe 1000_8 mots. Le background est amené en mémoire centrale par SCRUTF, et s'implante sur INTERF.

Le BACKGROUND est activé à partir de l'appel superviseur de la TTY Background. On peut alors taper les clés :

- ENB : fin de background
- NUL : détache et met hors tension les périphériques attachés au background
- RAS : monte en mémoire l'assembleur et permet de rentrer les clés assembleur
- FTR : monte en mémoire un module qui initialise la zone opérande FORTRAN

Sous la clé FTR, on a accès à deux types de clés FORTRAN :

- Modification de fichier :
  - . SCE
  - . BUT
  - . CRE
  - . MSG

. ERR
. CDE

- Clés modules :
  . ENF fin FORTRAN
  . FTF Initialisation Fortran
  . FCP Compilation
  . FXE Exécution Fortran
  . BIB Affectation de bibliothèque
  . FSG Sortie code généré
  . FDL Suppression d'une unité en bibliothèque
  . FLS Listing d'une bibliothèque
  . FEG Entrée de code généré
  . NUL Détache et met hors tension les périphériques

Les divers modules correspondants à ces clés sont sur disque et sont amenés en mémoire centrale à la suite de BOSF dans le premier bloc.

V.3 - Fichiers FORTRAN sur disque

Un utilisateur FORTRAN a à sa disposition des fichiers sur le disque sur lesquels il peut effectuer toutes les opérations classiques. Pour ce faire, il peut utiliser deux types de programmes :

- soit faire des Read/Write classiques en désignant comme numéro de périphérique un numéro spécial correspondant à ce qu'on appelle une US disque. La zone du disque ainsi accessible constitue un fichier séquentiel, en ce sens qu'à la suite d'un Read ou d'un WRITE, le prochain ordre d'entrée/sortie va lire ou écrire à la suite du dernier fichier lu ou écrit. On peut bien entendu faire un REWIND pour repositionner le pointeur au début de la zone disque,
- soit utiliser les sous-programmes
  . RFILW pour lire
  . WFILW pour écrire

  qui travaillent en demandant :
  . un numéro de fichier sur lequel on veut travailler
  . le nombre d'articles à sauter
  . le nombre de mots à transmettre
  . le nom du tableau où sont les données ou, où il faut implanter les données

  Après l'utilisation de RFILW ou de WFILW le pointeur du fichier correspondant est ramené obligatoirement au début.

Du point de vue système, tous ces divers types de fichiers utilisent SYSFIC. Dans le cas de l'US disque, qui est le même chiffre pour tous les utilisateurs, l'ordre d'entrée/sortie est piégé dans INTERF qui fait appel à MONE, et c'est dans FORES que la correspondance entre le fichier SYSFIC et l'US disque complétée par le numéro d'utilisateur est faite. FORES fait ensuite toutes les demandes à SYSFIC nécessaires pour la bonne marche de l'ordre d'E/S.

De même pour les programmes RFILW et WFILW, on établit une correspondance entre un fichier SYSFIC et le couple (numéro de fichier, numéro d'utilisateur).

V.4 - Conclusions sur le FORTRAN Temps Réel

Le FORTRAN Temps réel, tel qu'il existe dans le système CARINE, possède donc toutes les possibilités du FORTRAN autonome, c'est-à-dire toutes les possibilités classiques des FORTRAN existant sur les autres machines. Bien entendu, la dimension du mot (20 bits) limite la précision que l'on peut obtenir avec un tel outil : néanmoins, celle-ci est largement suffisante.

L'intervention du temps réel intervient au niveau des Entrées/Sorties, classiques ou spécialisées, qui peuvent mettre le programme en attente, ainsi qu'au niveau des arrêts, suspensions ou reprises, venant influer sur le déroulement normal d'un programme, en provenance de commandes utilisateurs ou de défauts externes provoquant des interruptions.

De plus, l'emploi du time-sharing pour permettre à plusieurs utilisateurs de travailler simultanément nécessite une gestion des transferts entre les divers programmes mémorisés sur disque et la mémoire centrale où on les amène pour les activer.

La mise en place du système FORTRAN sous RTDMS rend plus aisée l'optimisation du time sharing en désactivant tous les programmes faisant des E/S longues, dont MONES se charge, et qui ne sont donc pas obligés d'être présents.

Nous avons parlé, dans un chapitre précédent, des processus en général et nous avons vu que le programme qui contrôlait le processus pouvait être plus ou moins complexe, allant de la simple réponse aux interruptions à un ensemble de programmes imposant au processus une marche bien précise voulue par le programmeur. Le FORTRAN Temps réel est un bon exemple de ce deuxième cas : on a en effet le programme qui assure la gestion du processus (ici un appareillage de diffraction neutronique), dont la marche est donc entièrement programmable. Supervisant le tout, on a un deuxième système COSYNUS qui contrôle le déroulement des programmes FORTRAN eux-mêmes, et permet donc une grande souplesse par ce deuxième niveau de contrôle programmé.

VI - DESCRIPTION DE L'ENSEMBLE S'OFTWARE CHARGE DE LA GESTION DES ECHANGES AVEC LE CAMAC

Préliminaires

Nous rappellerons que cette partie a été réalisée par Monsieur DARIER, mais que nous avons inclu sa description dans le présent rapport, toujours dans l'optique de constituer un document complet sur le système CARINE.

VI.1 - Rappels

Nous avons vu plus haut que le système CARINE effectuait des échanges avec un système de transmission de données un peu spécial appelé CAMAC. La gestion de ces échanges a une très grande importance, car la bonne marche de l'appareillage, donc de l'expérience, y est intimement liée.

Le CAMAC est connecté au T 2000 par l'intermédiaire d'un Coupleur spécial, occupant une adresse bien déterminée dans le bac d'E/S, et dont le rôle principal est de réarranger les informations T 2000 pour les envoyer sur les lignes du CAMAC.

La gestion du CAMAC aurait donc pu être assurée par MONES puisque le CAMAC peut être considéré comme un périphérique du T 2000. Cependant, il a paru plus judicieux, à cause de la particularité de ces échanges, de créer un système software gérant le CAMAC et incorporé à RTDMS. On y gagne en rapidité, ce qui est parfois nécessaire dans certains échanges.

VI.2 - Description globale de l'ensemble

Le principal programme gérant le CAMAC est le moniteur d'E/S CAMAC, appelé MONAC, jouant un rôle comparable à MONES, et étant comme ce dernier, un classe 4 de RTDMS, donc accessible aux classes 2 utilisateurs.

A ce moniteur est adjoint un ensemble de programmes effectuant la liaison avec le FORTRAN, du point de vue instructions, ainsi que du point de vue gestion temps réel, car un échange CAMAC peut mettre un programme FORTRAN en attente longue.

Lorsque, dans l'interprêteur FORTRAN, on détecte un appel à un sous-programme FORTRAN, on transmet les arguments d'appel ainsi que le numéro de séquence à un programme de classe 2 appelé CPAS, qui est propre à chaque utilisateur. A partir de ce moment, la présence en mémoire centrale du programme FORTRAN utilisateur n'est plus nécessitée.

Le programme CPAS fait appel à un sous-programme réentrant, un classe 3 au sens RTDMS, qui assurera l'enchaînement des appels nécessaires aux tâches de classe 2 à activer ainsi que les appels à MONAC nécessités par les E/S.

Dans le cas d'une attente longue, CPAS est mis en attente. Il sera réactivé sur les appels caractéristiques de fin d'attente qui parviendront du module CAMAC adressé à travers MONAC et le module d'échange propre au module Hardware. Le schéma de la figure N° 17 résume ces divers points.

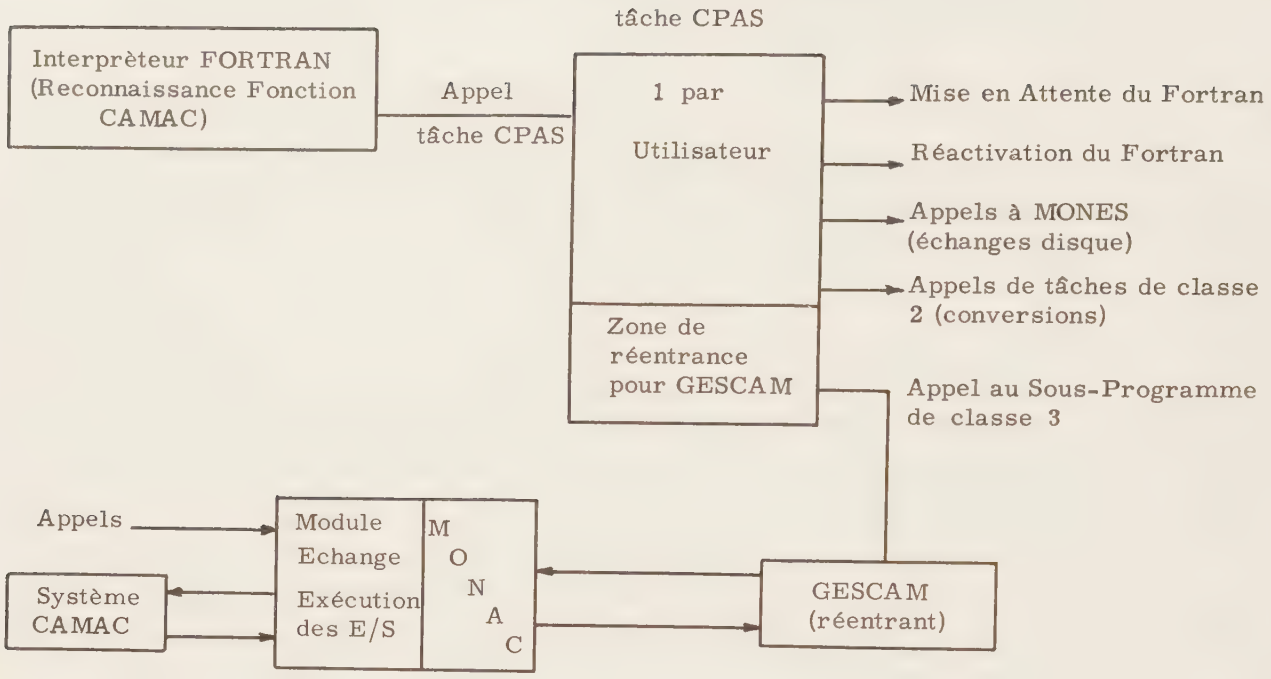


Figure N° 17

Vue d'ensemble du Système gérant le CAMAC

Le sous-programme GESCAM, qui est réentrant, utilise une zone de réentrance qui est dans le CPAS de l'utilisateur en cours de travail. C'est ce sous-programme qui effectue les aiguillages vers les modules spéciaux de la bibliothèque CAROLUS sélectionnés dans l'appel FORTRAN.

VI.3 - Liaisons FORTRAN-CAMAC

Tous les échanges avec le CAMAC se font par l'appel, dans le programme FORTRAN, d'un sous-programme d'une bibliothèque, appelée CAROLUS, (CAmac - ROutines Library for USers), lequel se chargera d'effectuer toutes les Entrées/Sorties avec l'appareillage des expériences à travers le CAMAC. Les sous-programmes de cette bibliothèque sont résidents et à la disposition de tous les utilisateurs.

On les active par un ordre :

CALL SP1 (P_1, P_2....)

Afin de faciliter la programmation, les divers axes mécaniques ou variables instrumentales,qui sont définies par l'adresse CAMAC des modules qui les gèrent, sont représentés par un Icode, qui est une unité symbolique FORTRAN. Les arguments d'appels des sous-programmes de la bibliothèque sont donc des noms symboliques FORTRAN ou des valeurs. Les noms symboliques représenteront donc un code adressant une variable physique

(Unité Symbolique FORTRAN) ou une valeur de consigne (cas d'un positionnement) ou le résultat d'une acquisition (cas d'une lecture).

La correspondance entre les Unités Symboliques FORTRAN et les modules Hardware est effectuée par des tables utilisées par le moniteur MONAC et pouvant être remises à jour par le système COSYNUS en cas de reconfiguration hardware de l'expérience.

De plus, chaque Unité Symbolique FORTRAN possède une table sur disque contenant :

- la butée basse pour la valeur
- la butée haute " " "
- la valeur du décalage de zéro
- la valeur de consigne

Ces tables peuvent être remises à jour, et avant de travailler sur un icode, une comparaison systématique est faite avec les deux valeurs de butée afin d'assurer la sécurité de fonctionnement de l'appareillage, et d'éviter les erreurs de frappe ou d'étourderie.

VI.4 - Description du moniteur d'E/S MONAC avec le CAMAC

Présentation

Le Moniteur d'E/S CAMAC MONAC est destiné à contrôler toutes les opérations d'entrées/Sorties entre le calculateur T 2000 et le standard CAMAC. Dans ce but, il assure tous les transferts d'information et le traitement des interruptions en provenance du système CAMAC.

Toutes les demandes au Moniteur se font à des unités Symboliques reliées aux stations physiques et à leurs adresses CAMAC par des tables propres au moniteur.

Lors de la configuration de MONAC, on peut définir plusieurs utilisateurs qui pourront lui faire appel en quasi-simultanéité.

Unités CAMAC

Les unités Symboliques utilisées lors de l'appel àMONAC seront appelées Unités CAMAC. Chaque Unité CAMAC représente de manière symbolique une adresse CAMAC.

On distingue deux types d'unités CAMAC :

- type 1 : ce sont les unités qui ne possèdent pas la possibilité d'appels ou dont les appels sont indépendants de l'utilisateur
- type 2 : ce sont les unités qui peuvent émettre un appel à la suite de certaines commandes.

Appel à MONAC

Un appel au Moniteur permet l'exécution d'un ordre CAMAC élémentaire ; si cet ordre doit provoquer un appel en retour, ce dernier est entièrement analysé par le moniteur, qui rend le contrôle à l'utilisateur en positionnant un compte-rendu, ou qui déclenche une action au niveau du système général (Arrêt automatique de programme, suspension...)

On distingue deux types de demandes de travail :

- les travaux courts : ce sont ceux qui ne provoquent pas la mise en attente du programme

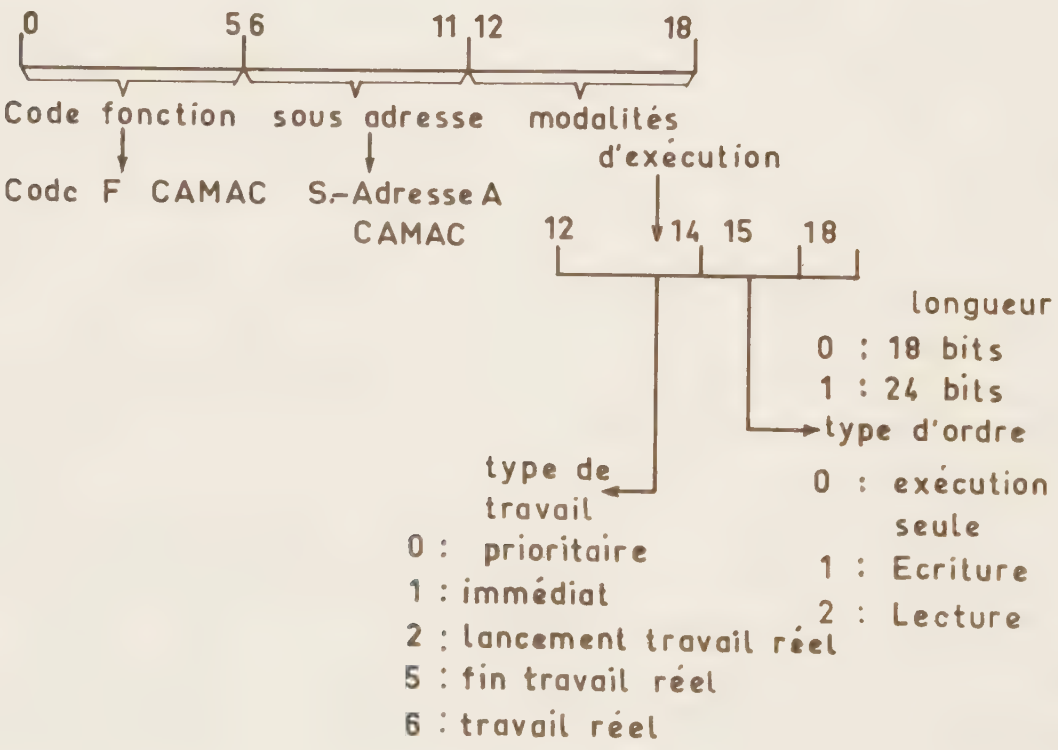
appelant, et il n'y a pas en général d'attente d'appel CAMAC pour acquitter l'ordre lancé.

- les travaux longs : ce sont ceux qui nécessitent la mise en attente du programme appelant, ce dernier n'étant réactivé que sur réception de l'Appel CAMAC correspondant. Il y a donc toujours attente d'un appel CAMAC en retour.

Toutes les demandes à MONAC sont de type séquentiel, le programme appelant ne reprenant la main qu'à la fin du travail demandé à MONAC.

Les modalités d'appels sont analogues à celle de MONES. En particulier, la table d'échange pour MONAC contient :

- nom de l'unité CAMAC : c'est le nom symbolique figurant dans la table de l'Unité CAMAC ; il s'agit de 2 caractères alphanumériques en représentation 2 caractères / mot interne au T 2000
- modalités d'appel : il s'agit d'un mot de 19 bits utilisés comme suit :



VI.5 - Comptabilités ETAT UC - TRAVAIL

Une unité Camac de type 1 ne possède pas d'état

Une UC de type 2 possède trois états :

- état occupation : indiquant que l'UC est en cours de travail et qu'elle va envoyer un appel
- état attente : indiquant que le programme appelant est en attente d'un appel
- état anomalie : signifiant un défaut léger sur l'unité CAMAC

Une unité de type 1 ne pouvant envoyer des appels consécutifs à une commande, on ne pourra lui demander que des travaux immédiats ou prioritaires.

Pour une unité de type 2, on a le tableau suivant :

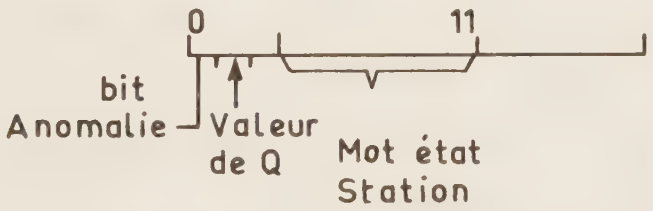
Etats		Type de Travaux autorisés
Occupation	Attente	
0	0	Tous
1	0	Travail prioritaire fin travail réel
1	1	Travail prioritaire

Compte-Rendu de MONAC

MONAC fournit un compte-rendu à l'utilisateur permettant de savoir si l'échange s'est déroulé correctement, ou quelle a été l'anomalie de fonctionnement.

La tâche appelante n'étant pas obligatoirement présente lors de la transmission du compte-rendu, celui-ci doit être résident.

Le compte-rendu a la configuration suivante :



La valeur de Q est celle transmise par le CAMAC à l'exécution de la fonction demandée au moniteur par l'utilisateur.

La présence du bit "Anomalie" signifie qu'un appel non attendu a été envoyé par l'unité CAMAC depuis le dernier ordre transmis à cette même unité .

Dans ce cas, l'échange demandé au moniteur ne s'est pas déroulé, mais l'unité CAMAC est à nouveau disponible pour un échange éventuel.

Le mot état station est nul si tout s'est déroulé correctement.

VI.6 - Traitement des Appels

Le Moniteur assure le traitement complet de l'Appel, sa remise à zéro et l'acquisition du mot d'appel s'il y a lieu. Lorsque l'Appel provenant d'une station trouve l'unité CAMAC correspondante en état occupation, cet appel est dit "synchrone", sinon, il est qualifié d'"asynchrone".

Après traitement et analyse, un appel peut être :

- un acquittement de bon fonctionnement

- une anomalie
- un défaut
- une alarme

L'action au niveau système est adaptée à la gravité de la panne, le choix étant fait dans le module d'échange propre à la station.

On a alors les actions suivantes :

- en cas d'alarme : arrêt du programme utilisateur
- en cas de défaut : suspension du programme
- en cas d'anomalie : positionnement du mot état station

Sur un appel asynchrone de bon fonctionnement ou d'anomalie, il y a toujours positionnement du bit anomalie dans le compte-rendu. L'Unité CAMAC est alors mise dans l'état défaut, état testé lors de la demande au moniteur suivant l'erreur asynchrone et sur la même UC. Dans ce cas la transmission est refusée, avec compte-rendu à l'utilisateur.

VI.7 - Tables de MONAC

MONAC travaille sur un certain nombre de tables, qui doivent être configurées selon l'installation hardware de l'utilisateur.

Ces tables donnent :

- les correspondances unités CAMAC Adresses réelles

  Ce sont les tables $TASAC_i$ (1 par utilisateur)

- les correspondances Niveau d'IT Unités CAMAC

  Ce sont les tables $TABIT_j$ (1 par niveau d'IT)

On trouvera en annexe la description de ces diverses tables. Notons cependant que, pour les tables TABIT, l'appel d'une station peut provenir :

- soit directement de la station par une IT
- soit à travers le chassis, l'IT chassis étant connectée directement au T 2000
- soit à travers le chassis et la branche CAMAC

Les tables TABIT comportent deux mots par sous-niveaux d'interruption et permettent de tenir compte de ces diverses possibilités d'appel.

VI.8 - Bibliothèque CAROLUS

Nous avons déjà mentionné plus haut cette bibliothèque qui permet d'entrer en relation avec le CAMAC à partir du FORTRAN et qui comporte des sous-programmes orientés vers la diffraction neutronique.

La composition de cette bibliothèque est la suivante :

(le signe * indique un tableau)

sous-programmés généraux

- LECT 1	(Icode, Theta)	lecture 1 variable
- LECTN	(Icode*, Theta*)	lecture n variables
- POS 1	(Icode, temps)	positionnement 1 variable
- POS 2	(Icode, Temps, Jcode, Theta)	positionnement de 2 variables
- POSN	(Icode*, Angle*)	positionnement de n variables

- COMP (Icode) Démarrage et Arrêt d'un comptage de type donné par Icode, a une valeur de consigne préfixée
- ACQ (Icode, Comp, Cx, Dx) Acquisition d'un point de mesure
- POSAC (Icode*, Angle*, Icode, Comp, Cx*, Dx*) Scan sur une ou deux variables couplées
- INIT : Initialisation du CAMAC
- CAMAC (KTRAV, IEX*, INF*) Transmission d'un ordre CAMAC élémentaire.

sous-programmes particuliers :

- ACM (Icode, Comp, Cx, Dx*) Acquisition d'un point de mesure sur un multidétecteur
- LEM (Icode, Cx*) Lecture d'un comptage sur n cellules du multidétecteur
- POMO (IPHI*, IRO*) Positionnement de 100 moteurs pour expérience D6
- COFILI (1*) Configuration du flipping

VI. 9 - Conclusions

L'ensemble de gestion du CAMAC constitue un outil très commode pour le physicien, pour gérer le déroulement de son expérience. La distinction faite entre le Moniteur d'Entrées-Sorties proprement dit, et l'interface avec le FORTRAN en rend l'utilisation très souple et permet toutes les adjonctions dans le futur.

Le traitement des Appels rend impossible tout "blocage" du système faute d'une réponse venant du CAMAC.

Enfin l'utilisation de tables et d'un adressage symbolique, facilitent le changement de configuration d'une expérience, par le minimum de modifications à faire.

VII. TESTS DES MODULES CAMAC PAR FORTRAN SOUS LE SYSTEME CARINE

Préliminaires :

Fort d'une expérience acquise précédemment sur le couplage d'un ordinateur CII 10010 au système CAMAC, nous avons étudié le problème du test des modules CAMAC, problème relativement important dans le cas du mauvais fonctionnement d'une ou de plusieurs expériences. Il faut en effet, pouvoir être sûr du fonctionnement des modules du système CAMAC, et cela sans perturber les autres expériences.

Le présent chapitre présente donc comment nous concevons ces tests dans le cadre du système CARINE.

VII.1 - Généralités

VII.1.1 - Comment tester un module CAMAC

Lorsque le standard CAMAC a été développé, la complexité de certains modules et les possibilités nombreuses de mauvais fonctionnement ont conduit à systématiser les tests effectués sur ces modules à l'aide d'ordinateurs.

Un module CAMAC HARDWARE, dans la philosophie d'un test, est une "boîte noire" qui réagit à certains ordres du calculateur

- adressés
- non adressés

par l'émission ou la réception de données ou de signaux de réponses pouvant être détectés soit par programme, soit "de VISU", par l'extinction ou l'allumage d'un témoin visuel quelconque.

C'est donc un ensemble d'équations logiques entre les signaux d'entrée et les signaux de sortie, lesquels peuvent être soit disponibles immédiatement au reçu de l'ordre, soit mémorisés et lus par la suite pour des éventuelles comparaisons.

En général, lors d'un test CAMAC, pour tester certaines fonctions, on est obligé d'en envoyer d'autres que l'on suppose a priori correctes. Donc, il est nécessaire en général de supposer a priori le bon fonctionnement de telle ou telle commande (la plus fiable) et, à partir d'elle, de tester toutes les autres systématiquement.

Pour cela, on prend le schéma logique du module CAMAC et on examine un par un quels sont les signaux émis en réponses aux ordres reçus.

On fait l'équation logique de chaque signal, et on établit toutes les combinaisons possibles des signaux d'entrées.

Ensuite, on voit s'il n'y a pas moyen de grouper certaines séquences afin de gagner du temps.

Pour les opérations effectuant des échanges de données avec le module, on envoie une donnée caractéristique à ce module, et on vient la relire par la suite. L'identité permet d'être sûr du bon fonctionnement des lignes de lecture et d'écriture correspondant aux bits significatifs de la donnée choisie.

VII. 1. 2 - Problèmes posés par les tests ON-LINE

Les tests ON-LINE doivent être compatibles avec le déroulement du système CARINE, c'est-à-dire :

- être contrôlés directement ou indirectement par RTDMS
- utiliser MONAC pour les E/S Camac
- utiliser MONES pour les E/S conventionnelles.

L'écriture des tests ON-LINE en ASMAT posait certains problèmes :

- MONAC étant conçu en liaison étroite avec CPAS, il y avait difficulté à utiliser les appels à MONAC sans alourdir ce dernier.
- Un module CAMAC en défaut peut générer des appels qui sont considérés en temps normal comme des alarmes graves par MONAC mais qui, en test, sont des appels normaux (ex : vérification des butées). De même, on peut attendre un appel qui n'arrive pas par suite de la panne sur le module, ce qui bloque la tâche en question mise en attente par RTDMS. Il faut alors prévoir une procédure spéciale pour la mettre en rappel.
- Dans les tests ON-LINE, il faut veiller à ne pas perturber les expériences des autres utilisateurs, c'est-à-dire vérifier toutes les adresses CAMAC fournies par l'utilisateur, afin de savoir si elles lui sont propres, ce qui alourdit le programme de test.

VII. 1. 3 - Ecriture des tests en FORTRAN

Nous avons été amenés à envisager l'écriture des Tests CAMAC en FORTRAN pour plusieurs raisons.

- Tout d'abord, il était prévu qu'on ne pourrait dérouler un programme FORTRAN expérience simultanément avec un Test. Donc, la zone Fortran utilisateur est disponible pour le Test et ne pénalise pas l'expérimentateur.
- La procédure de réactivation d'un programme FORTRAN mis en attente par une interruption qui ne vient pas est prévue.
- Le traitement des appels et alarmes est prévu.
- On peut disposer des programmes de la bibliothèque CAROLUS pour envoyer des ordres CAMAC à partir du FORTRAN.
- Le FORTRAN Télémécanique permet la manipulation de bits dans un mot.
- Pour certains traitements,le FORTRAN est plus souple que le langage machine.

Toutes ces raisons nous ont fait préférer l'écriture des Tests en FORTRAN à l'écriture en langage ASMAT.

Dans ce cas, les échanges avec le CAMAC sont contrôlés comme pour le déroulement d'une expérience par le CPAS de l'utilisateur.

VII. 1. 4 - Philosophie des tests "EN LIGNE"

- Tout d'abord, un test "en ligne" ne doit en aucun cas perturber le fonctionnement des autres expériences en cours.
- Un test ON-LINE ne constitue pas un examen détaillé du module permettant de localiser la panne. Il s'agit simplement de vérifier que le module fonctionne correctement lorsqu'il est sous le système CARINE.

- En conséquence, tous les tests possibles sur un module ne seront pas forcément effectués, soit parce qu'ils risquent de perturber les autres utilisateurs, soit parce qu'ils n'ont pas de signification sous CARINE.
- Les tests en ligne ne portent que sur la partie programmable du module. Tous les signaux ou commandes liés à un fonctionnement manuel du module seront ignorés. Par contre, ils pourront être testés par des tests en différés.

VII.1.5 - Liste des modules pouvant être testés ON-LINE

- D'après ce qui précède, on ne peut tester ON-LINE, aucun des modules appartenant au système proprement dit du CAMAC. De plus, comme la possibilités de plusieurs expériences par branche est envisagée, on ne peut tester le module COB.
- Certains modules présentent également peu d'intérêt pour les tests ON-LINE, s'ils sont seuls
  . le module GAM arrêt-marche
  . l'horloge
- Donc, dans un premier temps, nous nous intéresserons plus particulièrement au test des modules suivants :
  . échelles à présélection et chronométrie
  . commande Codeurs-Moteurs
  . Contrôleur de chassis

VII.2 - Problèmes posés par l'écriture des tests en FORTRAN

VII.2.1 - Utilisation de la bibliothèque CAROLUS

Le Sous-Programme le plus utilisé sera vraisemblablement le sous-programme CAMAC par :

CALL CAMAC (ICODE, KTRAV, INFO, LCONT)

où :

ICODE = 1 unité ASM MONAC (2 caractères α Numérique)
K TRAV = F, A, Type de Travail
INFO = Tableau de 2 mémoires pour info à émettre ou à recevoir
LCONT = Compte-Rendu de MONAC à l'Utilisateur

Les variables KTRAV, INFO et LCONT sont garnies ou utilisées par le programme lui-même. Par contre, le module de Test étant utilisé par tout le monde il faut pouvoir lui passer le code de l'US ASM sur laquelle il doit travailler.

L'US ASM est composée de 2 caractères α Numériques :

- le 1° caractère est une lettre caractéristique du module CAMAC
  . E pour une échelle
  . M pour un moteur
  etc.....
- le 2° caractère est un chiffre qui désigne le n° de l'échelle pour un utilisateur qui en a plusieurs

par exemple : 1 pour moniteur
2 pour prétemps
3 pour acquisition

Le module de test s'adressant à un même type de module CAMAC, le 1° caractère, c'est-à-dire la lettre est figée. Seul le n° peut varier (le n° de l'utilisateur est transmis par le système).

En conséquence, pour un module de Test, il faudra faire entrer sur la TTY utilisateur le chiffre correspondant au 2° caractère de l'US ASM. Ceci est possible par un
- READ avec Format A

Pour les tests Moteurs-Codeurs, il faudra demander à l'utilisateur d'indiquer les voies qu'il veut voir tester. On pourra ainsi utiliser les sous-programmes
- POS, LECT et ACQ

VII. 2. 2 - Découpage en temps des programmes FORTRAN

Lorsqu'on fait plusieurs CALL CAMAC consécutifs, il faut songer qu'il peut se passer un temps assez long (théoriquement 24 s) entre les 2 CALL. Si les 2 CALL n'ont pas de rapport entre eux, ce n'est pas gênant, mais s'ils sont liés il faudra prendre toutes les précautions afin que dans le plus mauvais cas, le programme fonctionne correctement.

VII. 2. 3 - Problèmes particuliers

Pour l'échelle, lorsqu'on fait l'ordre Z, celle-ci démarre et peut générer une interruption. Or, pour envoyer Z, il faut s'adresser à une US ASM appelée "Z 1" et qui n'est pas prévue dans MONAC avec gestion d'appel. Or, ici l'appel est le signe que l'échelle a bien fonctionné.

On peut tourner le problème par la séquence suivante :
- demande de lancement de Travail réel à MONAC qui arme les appels sur l'US ASM de l'échelle,
- demande de travail immédiat qui arme Z et désarme I ,
- demande de mise en attente à MONAC

Si l'IT arrive on réactive alors le programme FORTRAN. Si elle n'arrive pas, c'est que le test est mauvais.

VII. 2. 4 - Incidents possibles - Résultats du test

Sur certains modules, c'est l'Appel généré en retour à une commande qui indique que le test est correct. Si le module incriminé est mauvais, cet Appel ne revient pas, et le programme FORTRAN n'est pas réactivé. On ne peut le signaler à l'utilisateur.

Ce dernier s'apercevra du non fonctionnement du test par le fait que pendant un certain temps rien ne se passe.

De même, s'il y a un défaut ou une alarme pendant le déroulement du test, le programme FORTRAN sera suspendu.

Dans tous les cas, l'utilisateur pourra
- l'arrêter par SFP
- le reprendre par RSP

Le test d'un module est constitué par une série de petits tests successifs. Pour indiquer à l'utilisateur que le programme de test se déroule bien, à la fin de chaque test partiel, on imprimera sur sa TTY un message de compte-rendu disant si le test est bon ou mauvais. L'absence de message pendant un certain temps indiquera au physicien que le test s'est bloqué quelque part.

VII.3 - Test échelle ON-LINE

VII.3.1 - Montage HARD nécessaire

Ce test nécessite de relier l'échelle à tester à l'horloge, sur une sortie dont la fréquence reste à préciser. Il faut que cette fréquence soit suffisamment élevée pour qu'on puisse saturer l'échelle assez rapidement (quelques secondes) et suffisamment lente pour qu'on n'ait pas le temps d'arriver à saturation entre 2 CALL CAMAC successifs. Une fréquence de 100 kHZ semble convenir, les essais devant la préciser par la suite.

L'échelle à tester reste au même emplacement qu'en fonctionnement normal. Si on la déplace, avant de faire le test, définir la nouvelle adresse par la commande HSC.

VII.3.2 - Paramètres à entrer

Au début du programme, on demande par la TTY utilisateur que ce dernier donne le numéro de l'échelle qu'il veut tester. Ce chiffre est celui qui figure dans l'US ASM utilisée par MONAC.

Ex. : E 1 , E 2 etc...

C'est le seul paramètre nécessaire au déroulement du test. Le numéro de l'utilisateur est fourni directement par le système qui connaît quel est le programme Fortran qui est en train de se dérouler.

VII.3.3 - Sous-Programmes CAROLUS utilisés

Le seul sous-programme CAROLUS utilisé dans ce test est le sous-programme CAMAC (Icode, KTRAV, INFO, ICONT)

Les US ASM utilisés sont :

- celle de l'échelle à tester : E 1 par exemple
- l'US ASM : C_o qui correspond à N = 30 et qui permet de générer Z, C et I

VII.3.4 - Tests effectués sur l'échelle

Pour ces tests, voir l'organigramme détaillé qui est joint. Sur cet organigramme figurent en marge des "Notes" que nous allons expliciter ci-dessous.

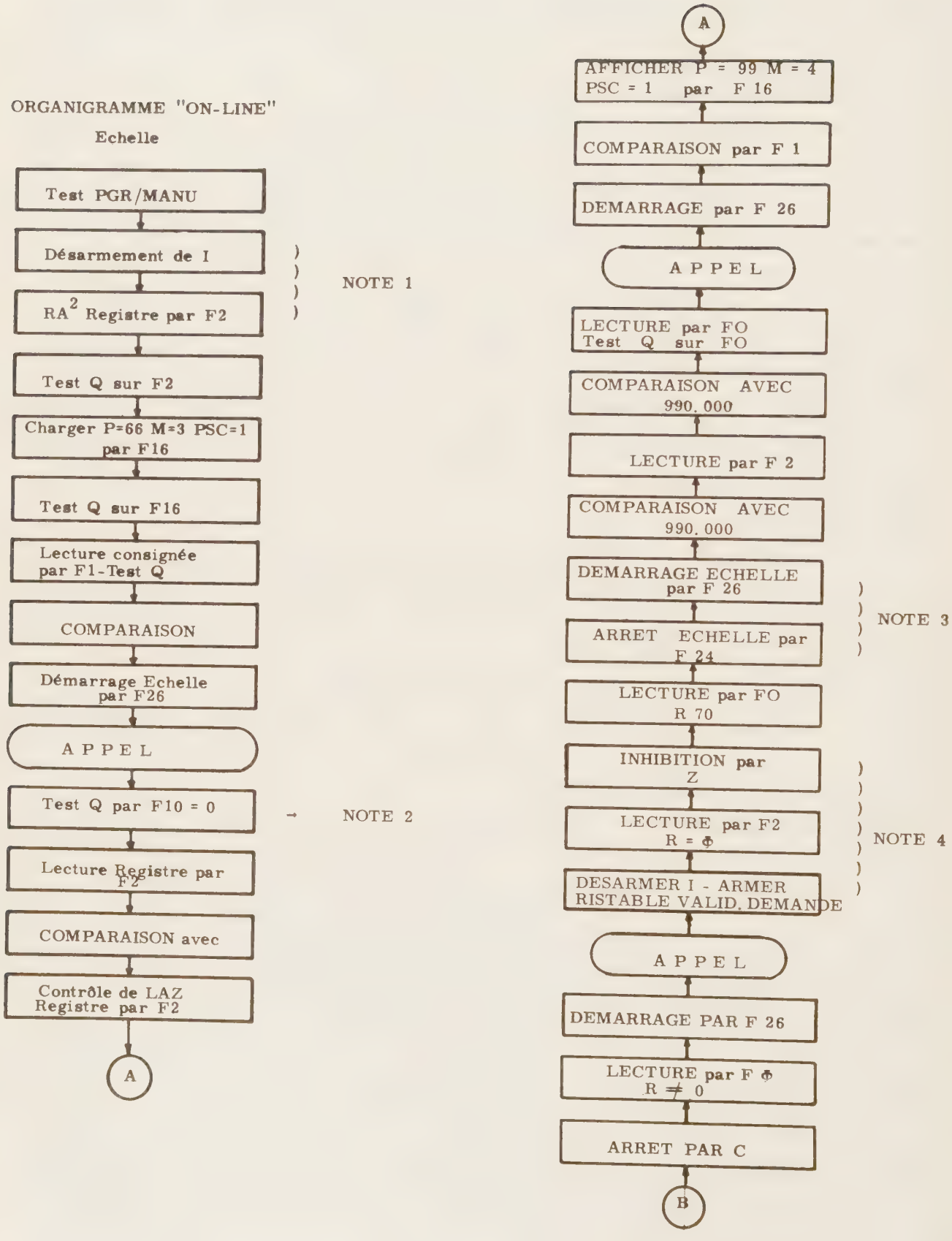
Note 1 : ces tests constituent une sécurité qui permet d'être sûr que l'échelle est bien initialisée.

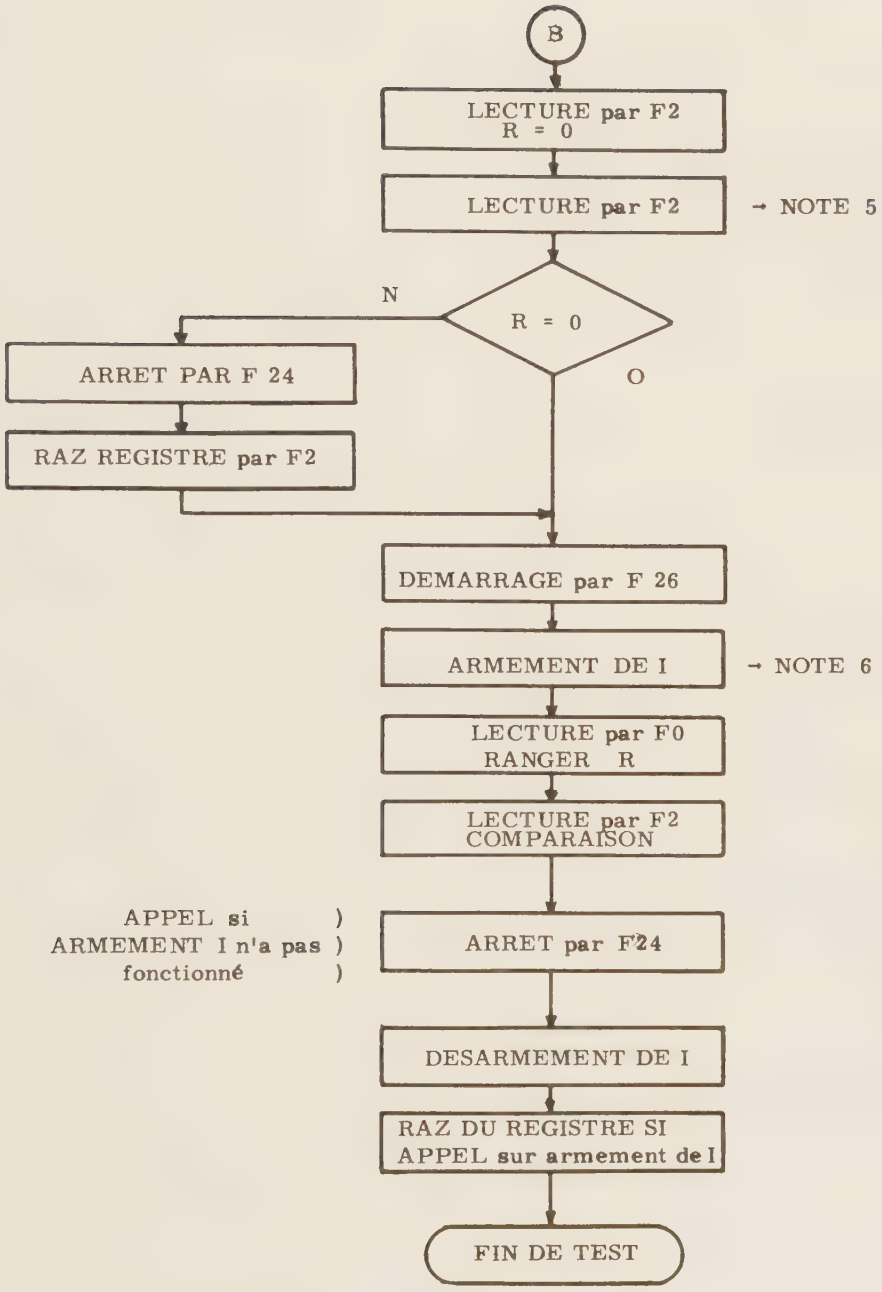
Note 2 : ce test peut être superflu, car en principe, MONAC remet à zéro l'Appel une fois qu'il l'a pris en compte. Ce test permet cependant de vérifier le bon fonctionnement de MONAC à ce point de vue.

Note 3 : Il se peut qu'entre le lancement de ces deux ordres F (26) et F (24), il se passe beaucoup de temps et que l'appel ait le temps de monter. On peut se prémunir en mettant une fréquence faible à l'entrée de l'horloge. De toutes façons, après le

ORGANIGRAMME "ON-LINE"

Echelle




B
LECTURE par F2
R = 0
LECTURE par F2
→ NOTE 5
N
R = 0
O
ARRET PAR F 24
RAZ REGISTRE par F2
DEMARRAGE par F 26
ARMEMENT DE I
→ NOTE 6
LECTURE par F0
RANGER R
LECTURE par F2
COMPARAISON
APPEL si
ARMEMENT I n'a pas
fonctionné
ARRET par F24
DESARMEMENT DE I
RAZ DU REGISTRE SI
APPEL sur armement de I
FIN DE TEST

F (24), on fera un CALL CAMAC ineffectif sur l'échelle pour voir dans le compte-rendu s'il y a eu appel ou non. S'il y a eu appel, on recommence le test car, on est peut-être tombé sur un cas défavorable. Si ce test boucle un certain nombre de fois (10 par ex.), à ce moment-là on considère qu'il y a erreur et que l'arrêt par F 24 ne marche pas.

Note 4 Ici on veut armer l'échelle par Z et attendre un appel.
Or, l'US ASM correspondant à Z n'est pas équipée d'une gestion des appels. Pour remédier à cela, on fait la séquence suivante :
- demande de lancement de travail réel sur US Echelle qui arme le système d'Appel dans Monac.
- exécution des CALL CAMAC correspondant à Z et "désarmer I"
- demande de mise en attente qui mettra le programme en attente si l'appel n'est pas arrivé ; si l'appel est là, on réactivera immédiatement le programme.

Note 5 Ici, si l'arrêt par C ne fonctionne pas, on va avoir un appel non attendu. Il faut prévoir un CALL CAMAC ineffectif pour voir si on a eu un appel afin de réinitialiser l'US ASM de l'échelle.

Note 6 Ici, analogie avec les notes 2 et 5. Si l'armement de I ne fonctionne pas, ou si on a un délai trop long entre les 2 CALL CAMAC, on va avoir un appel non attendu. Procéder comme pour les notes 2 et 5.

VII. 4 - Test ensemble CODEURS-MOTEURS

VII. 4. 1 - Paramètres à entrer

Dans ce test, l'utilisateur peut tester 1 ou plusieurs ensembles moteurs-codeurs. Donc, pour travailler, ce sous-programme a besoin de connaître :
- le numéro de l'utilisateur (ou son code identification)
- la liste des moteurs qu'il veut tester. Il doit donner les ICODE correspondant à ses moteurs, tels qu'ils sont définis dans CAROLUS.

Grâce à des tables, le module de test saura quelles sont les US ASM correspondantes. Donc, ce test commence par une demande clavier à l'utilisateur, où ce dernier indique les paramètres énoncés plus haut.

VII. 4. 2 - Sous-programmes CAROLUS utilisés

Ce sous-programme de Test utilise les sous-programmes CAROLUS suivants :
- CAMAC : pour l'envoi d'ordres élémentaires CAMAC
- POS : pour le positionnement des moteurs
- LECT : pour la lecture des codeurs.

VII. 4. 3 - Tests proprement dits effectués

* Au début, on demande donc à l'utilisateur :
  - de donner son numéro
  - de fournir la liste des voies à tester

* Puis, on lui demande d'aller relever en Manuel les valeurs actuelles des codeurs et de les noter sur une feuille pour pouvoir comparer par la suite à la valeur lue par programme.

* Test du GCC10 Codeur

  - 1) test de l'état PRG - Manuel par F (27) avec réponse sur Q
  - 2) test des fonctions ayant une réponse sur Q, c'est-à-dire :

    F (0)
    F (10)
    F (12)
    F (16)
    F (17)
    F (27)

  - 3) test du registre de lecture pour savoir s'il n'y a pas de 1 présents

* Test plaquette Codeurs

  - 1) test sélection codeurs et mesures

    On sélectionne toutes les voies indiquées par l'utilisateur, avec lecture de ces voies et impression sur la TTY. Le physicien peut alors comparer ces résultats avec les valeurs relevées en manuel auparavant.

  - 2) test inhibition : on vérifie qu'on ne peut avoir 2 sélections ext. successives sans une RAZ préalable de la 1°. Ce test permet de vérifier la bonne marche des codes F (9) et F (11)
  - 3) Test RAZ par C : on vérifie que C effectue bien les RAZ prévues

* Test GCC10 Moteur :

  Les tests sont identiques aux tests du GCC10 Codeur

* Test Plaquette Moteur

  - 1) Test registre Ro : on vérifie que l'on a bien RO = 0
  - 2) Test de la commande Marche Horloge
  - 3) Test du bit fonction d'un moteur
  - 4) Test du bit fin positionnement d'un moteur
  - 5) Test inhibition : on ne peut charger le registre Nb de pas si ce dernier n'est pas nul.
  - 6) Test chargement registre de sens pour un moteur
    - sens avant
    - sens arrière
  - 7) Positionnement du moteur dont la valeur de butée haute est la plus élevée à la valeur maximale
    - une première fois à 999...9 (autant de chiffres 9 que permis sans aller en butée)
    - une fois à 666...6

Ce test permet de tester tous les bits. Si ce test est incorrect, on ne peut savoir si c'est l'ensemble moteur ou l'ensemble codeur qui est défectueux.
FIN du test.

VII. 5 - Test du CRC

VII. 5. 1 - Généralités

Les tests ON-LINE du CRC sont des tests effectués dans un chassis CAMAC normal et non susceptibles de générer des Appels. En effet, les US ASM utilisés dans MONAC pour travailler sur le CRC ne sont pas équipées d'un système de gestion des Appels.

Une expérience pouvant comporter plusieurs chassis, l'utilisateur devra indiquer quel est le numéro du chassis à tester. Il ne s'agit pas du numéro choisi sur la face avant du CRC, mais du numéro dans l'expérience.

VII. 5. 2 - Montage HARD nécessaire

Le test du CRC nécessite la présence dans le chassis d'un module "répondeur" enfiché à l'adresse 08. Si cette adresse devait être changée, faire la modification nécessaire par la commande HSC de COSYNUS.

Tous les autres modules CAMAC du chassis doivent être défichés afin de ne pas perturber le test.

VII. 5. 3 - Sous-Programmes CAROLUS utilisés

Ce sous-programme de test utilise le sous-programme CAMAC tel qu'il a été défini plus haut.

Les US ASM sur lesquels ce programme va travailler sont :

- C 1 pour N = 30
- Z 1 pour N = 28
- R 1 pour le répondeur
- une US ASM qcq pour les tests de F, A, etc...
  par exemple E 1

VII. 5. 4 - Tests effectués

Ce sous-programme comporte :

- test des lignes F
- test des lignes A
- test des lignes R et W
- test de Z
- test de C
- test de I
- test du bistable de validation de la demande

VII. 5. 5 - Précautions éventuelles

Il n'y a pas de précaution à prendre, sinon d'être certains qu'aucun des tests effectués ne génère d'appel.

VII. 6 - Sous-Programmes généraux communs

Les tests effectués ayant des structures et des séquences assez différentes, le nombre de sous-programmes communs est assez limité.

Cependant on peut envisager

- un S-P ARANGE appelé par

CALL ARANGE (KF, KA, KT, KC, KTRAV)

qui, à partir de

- un code fonction KF
- une s - A KA
- un type de travail KT
- une commande KC

donne une configuration dans un mot K TRAV prête à être utilisée par le SP CAROLUS CAMAC

KF = code fonction de 0 à 31

KT = type de travail 0 travail prioritaire
1 travail immédiat
2 lanc^{t} travail
5 mise en attente
6 travail réel

KC = commande 0 commande
1 écriture 1 mot
2 écriture 2 mots
3 lecture 1 mot
4 lecture 2 mots

- un S-P JOCKEY appelé par

CALL JOCKEY (I)

qui imprime OK
Valeur de I

- un S-P DEFAUT appelé par

CALL DEFAUT (I)

qui imprime NOK
Valeur de I

CONCLUSION

Nous nous sommes attachés, dans le présent rapport, à décrire le système qui pilote les expériences de diffraction neutronique du Réacteur à Haut Flux de l'INSTITUT LAUE-LANGEVIN.

Nous avons essayé de le faire en présentant la chose dans sa globalité , c'est-à-dire en essayant de faire saisir quels étaient les problèmes à résoudre, et de quel matériau on disposait.

Il ne faut pas perdre de vue que cette réalisation a été faite avec un certain produit informatique, tant Hardware que Software, et qu'on a voulu utiliser toutes ses ressources, afin d'aller le plus vite possible et de ne pas réécrire des programmes déjà existants.

C'est pourquoi, il pourrait paraître a priori, qu'un tel système soit un peu lourd, et qu'au lieu d'utiliser le FORTRAN, on aurait pu créer quelque chose d'entièrement nouveau. Mais, il ne faut pas oublier que le FORTRAN paraissait très utile comme langage de base, et que les seuls problèmes étaient ceux d'un système de commande et de la gestion du CAMAC.

Le système CARINE a donc été réalisé en tenant compte des impératifs de sécurité de fonctionnement, et de minimisation de place en mémoire centrale, tout en offrant le maximum de possibilités aux utilisateurs (8K de FORTRAN soit quelques 800 instructions sans compter les sous-programmes).

Bien entendu, ce gain de place a été rendu possible par le fait que, très souvent, les mesures sont longues (à l'échelle de l'instruction machine) et que, donc pour un utilisateur donné, on n'est pas trop pressé par le temps en général. On a donc pu concevoir un système où une grosse partie des programmes se trouve sur disque, et où les échanges entre le disque et la mémoire sont particulièrement fréquents.

Le système CARINE répond aux exigences que l'on s'était fixées au départ et, jusqu'à présent, a donné toute satisfaction. Sa conception, qui utilise un FORTRAN temps réel avec une interface avec le système CAMAC, en fait un produit original et, à notre connaissance, c'est une des premières réalisations utilisant un système CAMAC piloté par un langage évolué.

A l'heure actuelle, avec le développement des mini-ordinateurs, le système aurait peut-être été conçu différemment. Il aurait été plus simple de connecter un petit ordinateur par expérience, effectuant l'acquisition et le traitement, et de relier les diverses expériences à un ordinateur plus gros, utilisé en calcul scientifique. Un tel système serait plus souple, car les expériences seraient vraiment indépendantes et simultanées, et d'autre part, avec les progrès faits dans l'utilisation de la microprogrammation, les petits ordina-

teurs pourraient posséder un jeu d'instructions adaptées spécialement à l'expérience.

Le système CARINE se place néanmoins parmi les systèmes à la fois les plus souples et les plus sûrs, qui sont deux qualités difficiles à concilier.

BIBLIOGRAPHIE

[1] - DOCUMENTS RELATIFS A LA DIFFRACTION NEUTRONIQUE :

William R. BUSING and Henri A. LEVY

"Angle Calculations for 3 and 4 circle X-RAY and Neutron diffractometers"

Acta Crystallographica (1967), 22, 457.

[2] - DOCUMENTS RELATIFS AU SYSTEME HARDWARE

Brochures Télémécaniques sur le T 2000

- Unité centrale
- Entrées/Sorties

C. LALLEMANT et M. SARQUIZ

"Système CAMAC"

Bulletin d'Instrumentation Nucléaire, n° 37 - juin 1969 - pp. 33-41

"CAMAC, système d'instrumentation modulaire pour le traitement des informations".

Bulletin de l'Euratom - mars 1969.

[3] - AUTRES REALISATIONS SUR LE MEME PROBLEME

R. BOUCHE

"Langage temps réel TROL"

Communication présentée au Congrès International d'Electronique Nucléaire de Versailles - septembre 1968.

J. SCHOEFFLER et R. TEMPLE

"A real-time language for industrial process control"

Proceedings of the IEE - Vol. n° 1 - janvier 1970 - pp. 98-111

Herbert PIKE

"Process control software"

Proceedings of the IEE - vol. 58 - n° 1 - janvier 1970 - pp. 87-97

G. BASSI - M. GEYNET - J.C. GUITTEL

"Langage algorithmique de mesures d'intensité"

Note technique DN 70/1 CENG - janvier 1970

J. LECLERC

"Langage temps réel AUTOL"

Communication présentée aux Journées de l'Automatisme dans la Recherche et l'Industrie Nucléaire - GRENOBLE, juin 1969

Real L'ARCHEVEQUE

"Système SUCCESS"

Communication présentée au colloque international sur l'Electronique Nucléaire à VERSAILLES - septembre 1968.

E. E. MASLIN - S. H. TERRY - L. A. WRAIGHT

"Use of a PDP-8 computer for operation of several neutron beam experiments"

Nuclear Instruments and Methods - 71 - (1969) pp. 13-26

[4] - DOCUMENTS RELATIFS AU SYSTEME CARINE DU POINT DE VUE SOFTWARE

- Brochure Télémécanique "Software T 2000"
- "FORTRAN Temps Réel" Module ST. 2203 Réf. 1/62 203/00 30 00 de Télémécanique
- A. BARTHELEMY et G. PASTOR

  "Notice d'utilisation de la console de visualisation"

  Note ST - GP/ml - 71/1092 Institut LAUE-LANGEVIN

A. BECCIA

"Système CARINE - Bibliothèque FORCAM"

Note LETI/MCTE n° 72/2563

M. DARIER - ENTRESSANGLE - LESOURNE - VINIT - LANINO - POUSSIN - BECCIA - ROURE - BOUVIER

"Système CARINE" - Spécification"

Note LETI/MCTE - n° 72/287

M. LANINO - ROURE - TROTTOT

"Test du système CAMAC"

Note LETI/MCTE n° 1854

A. BECCIA

"Tests automatiques programmés des échelles JEP-20, JEP-10, FHC 1301-A par calculateur T 2000é

Note LETI/MCTE n° 1481

G. ENTRESSANGLE - LEDEBT

"Organisation des blocs de données sur disque"

Note LETI/MCTE 71/1675

Manuscrit reçu le 12 mars 1973

TABLE DES ANNEXES

ANNEXE I - Organigrammes de principe des tâches de COSYNUS

ANNEXE II - Organigrammes de principe des tâches gérant le DISPLAY

ANNEXE III - Documents relatifs au système SYSFOR

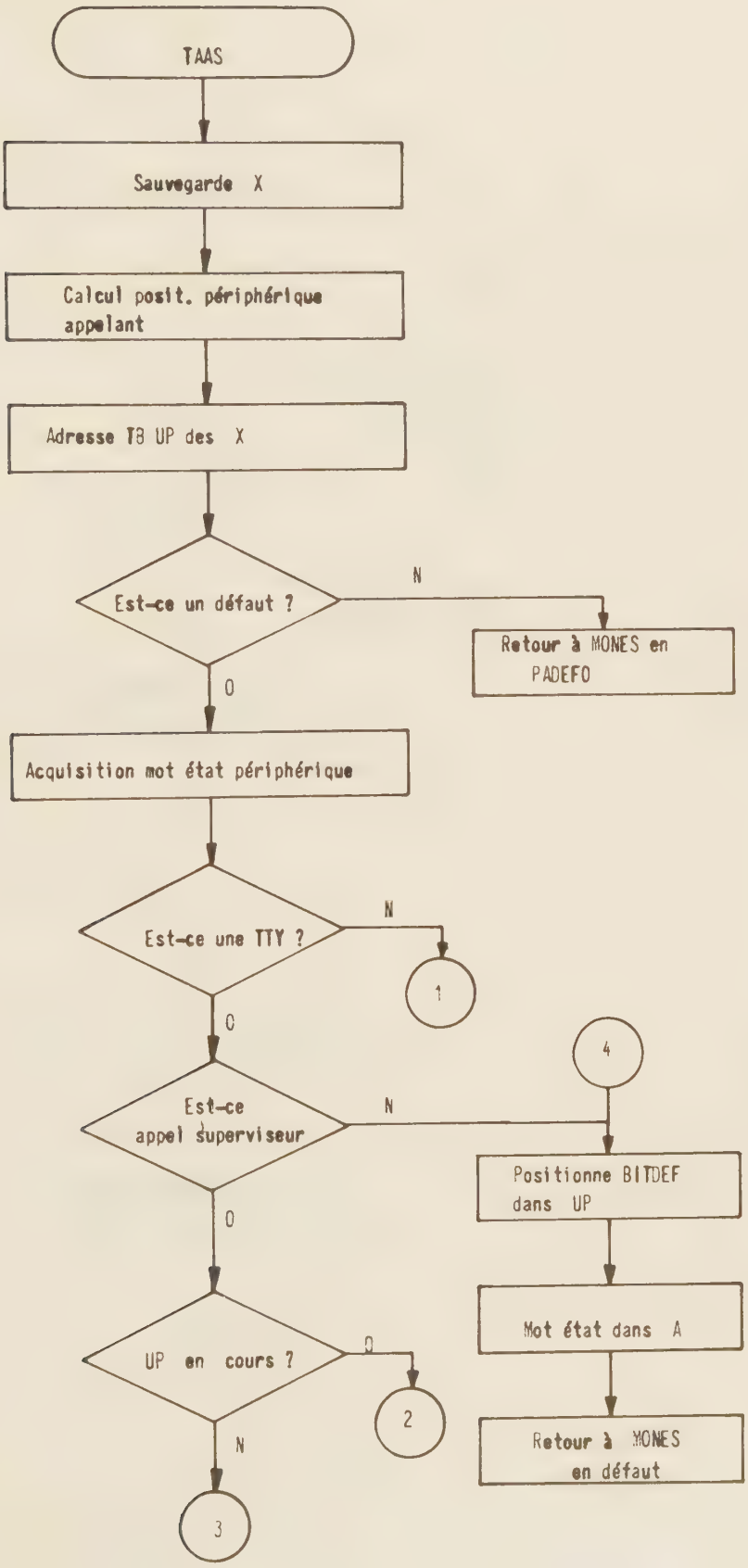
ANNEXE IV - Tables utilisées par le système gérant le CAMAC

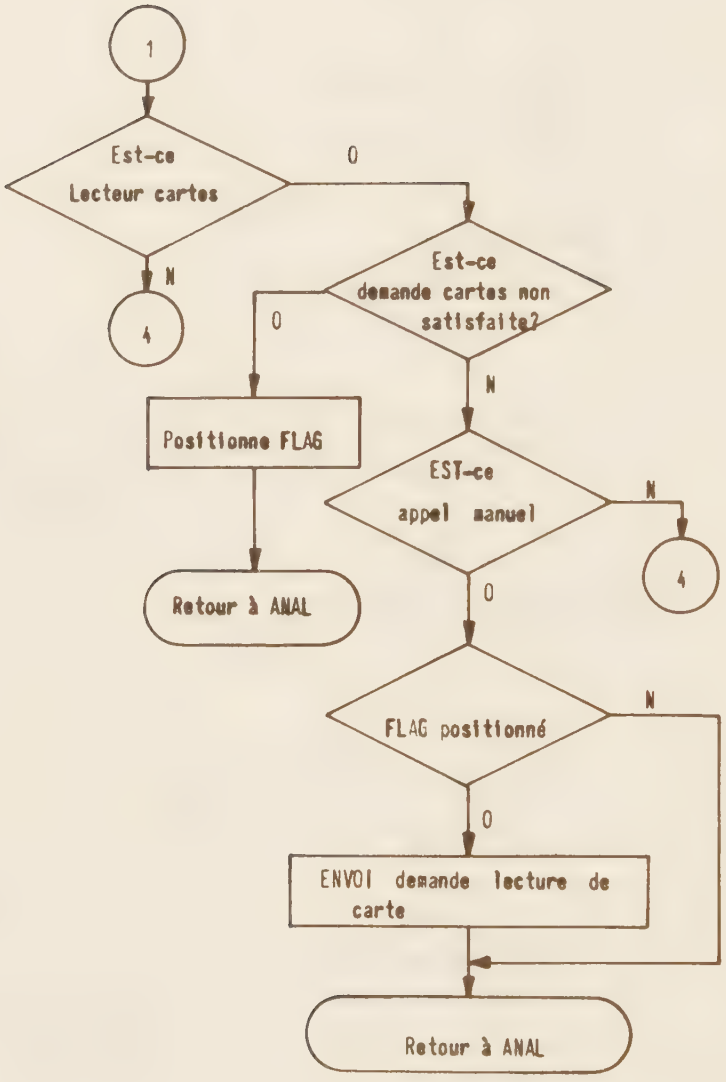
ANNEXE I

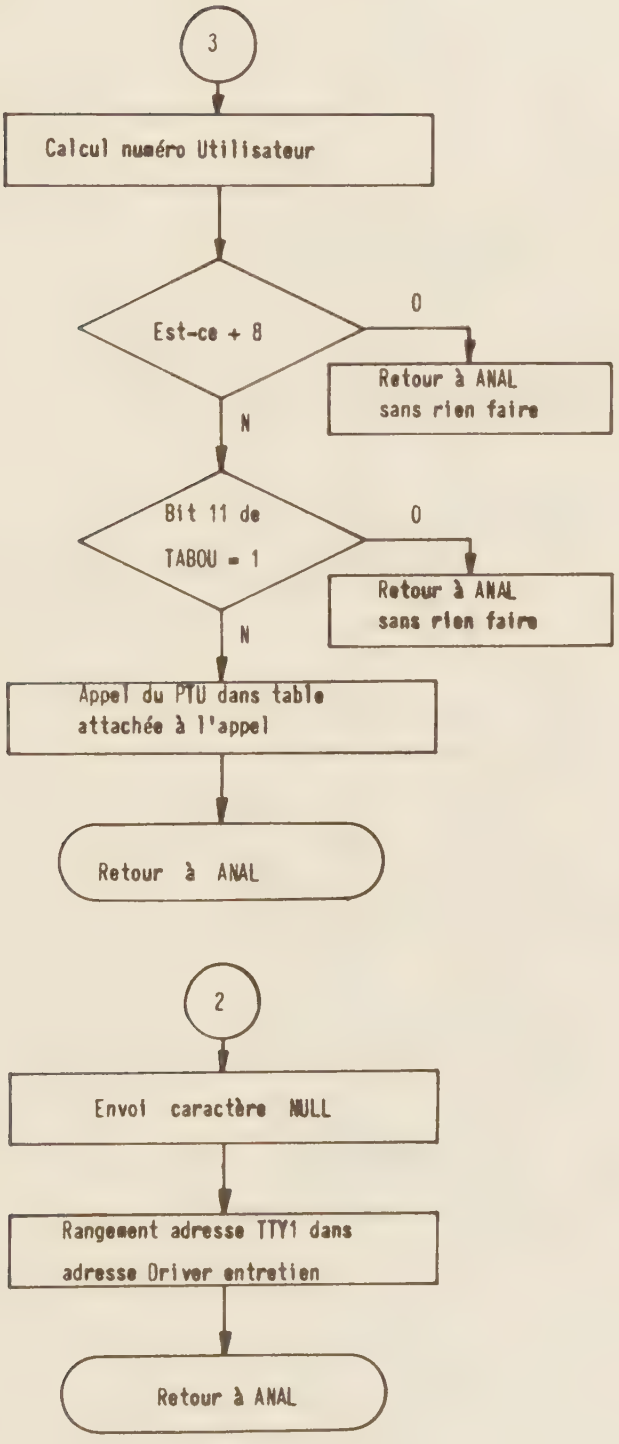
ORGANIGRAMMES DE PRINCIPE

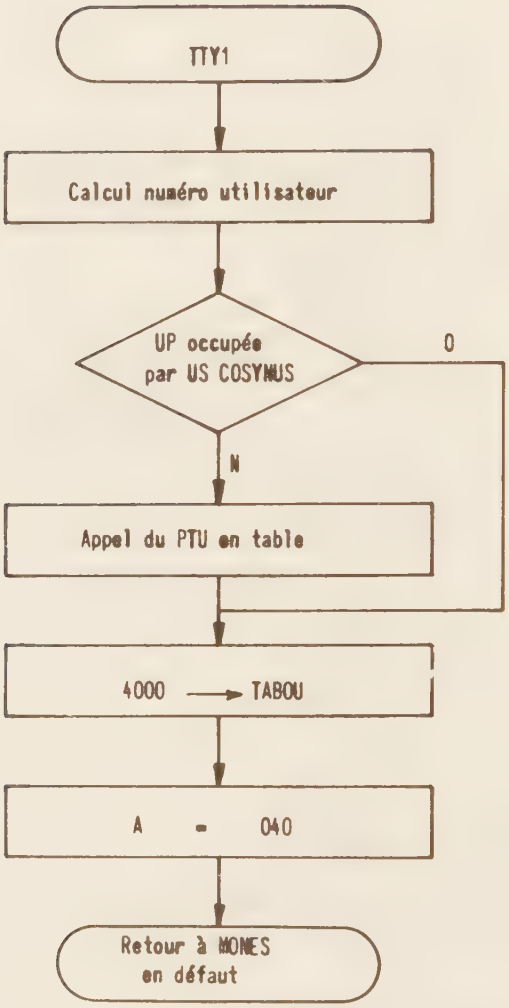
DES TACHES DE COSYNUS

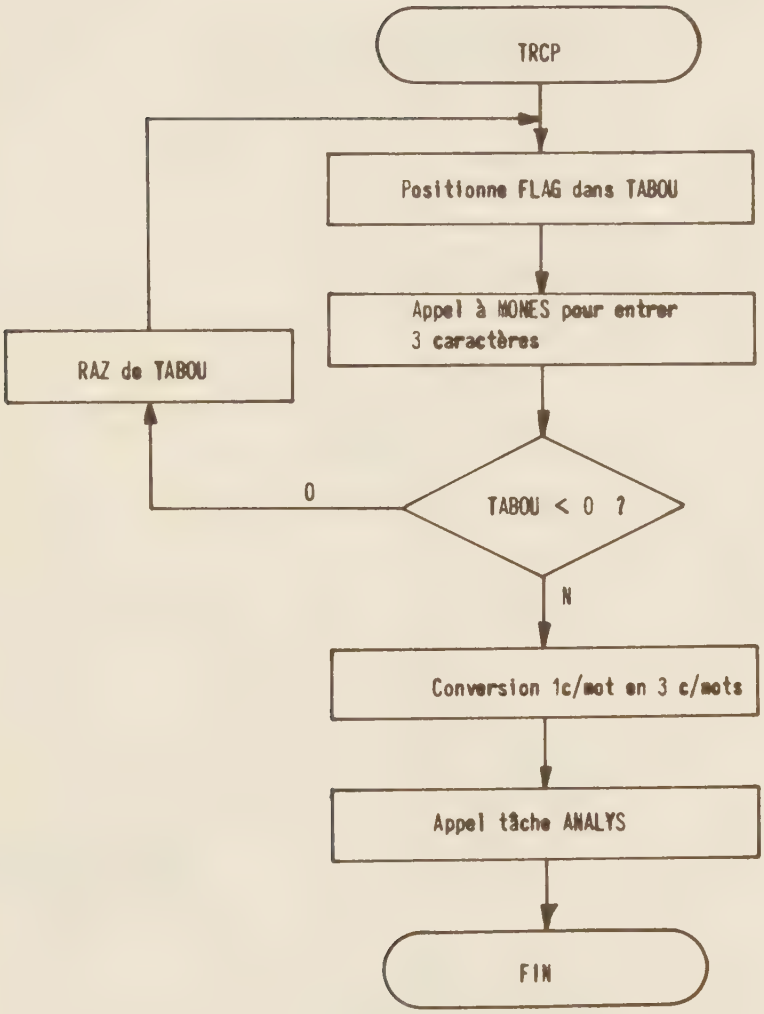
TRAITEMENT DES INTERRUPTIONS DU BAC E/S

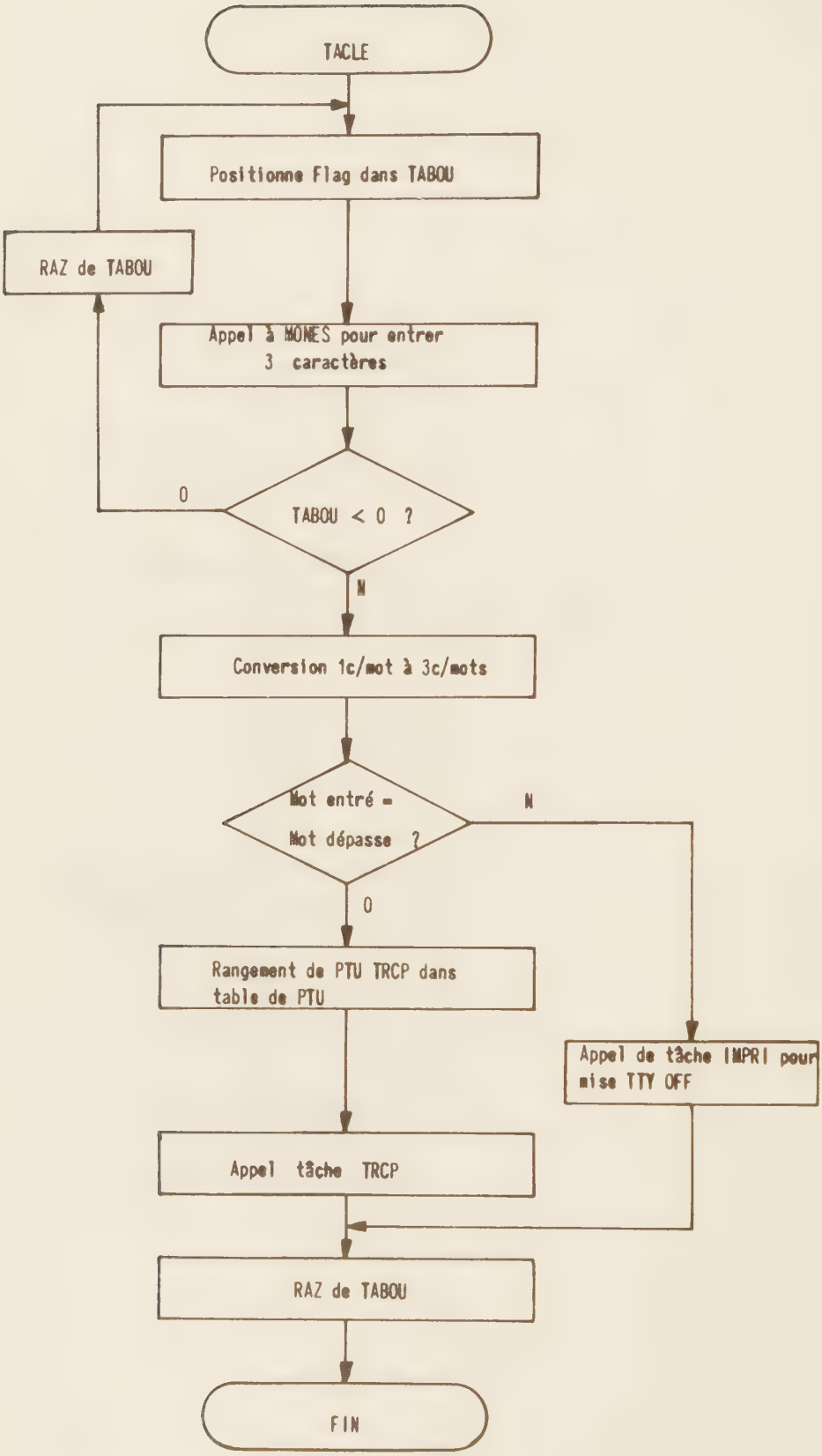


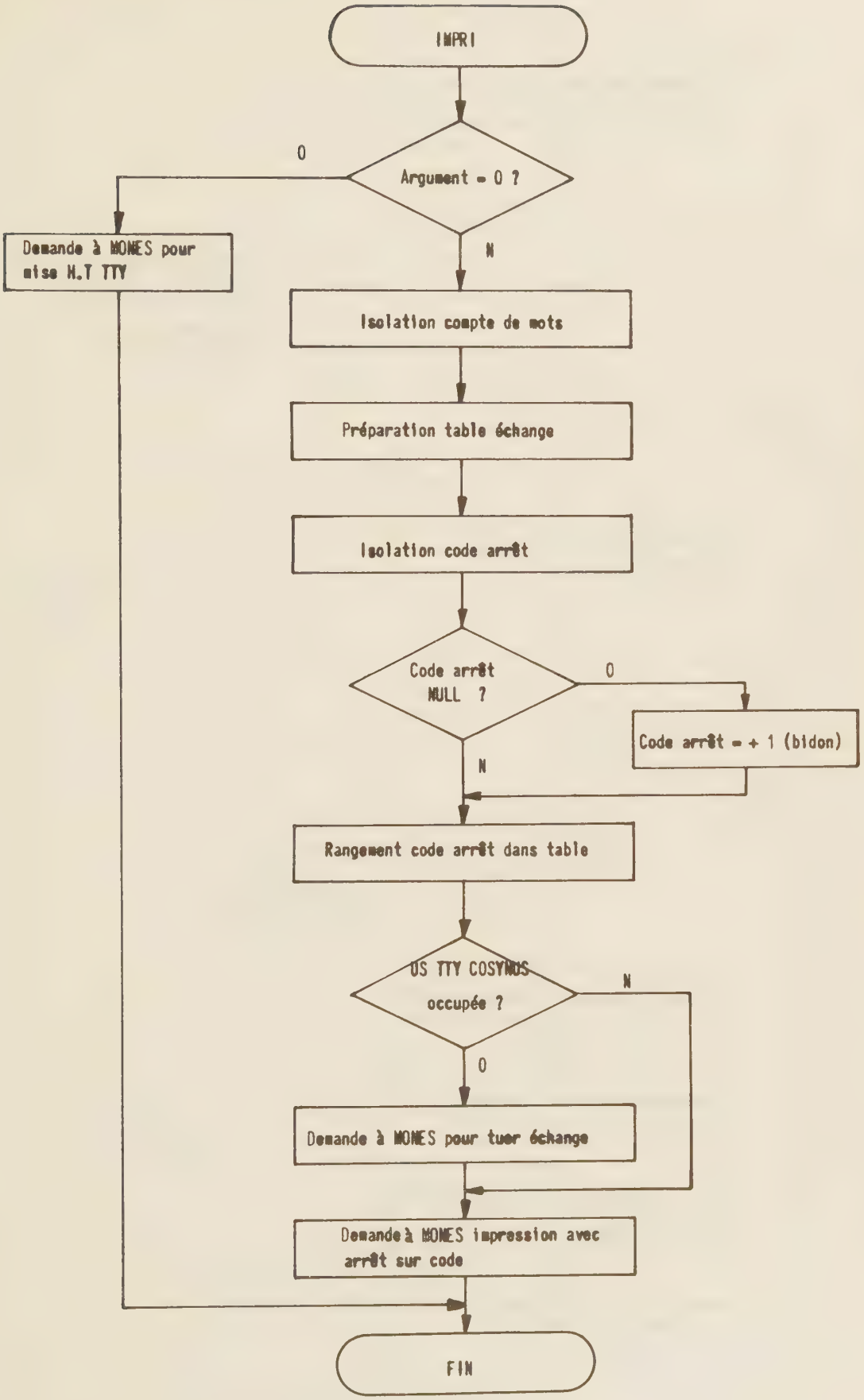

1
Est-ce
Lecteur cartes
O
N
4
Est-ce
demande cartes non
satisfaite?
O
N
Positionne FLAG
Retour à ANAL
EST-ce
appel manuel
N
4
O
FLAG positionné
N
O
ENVOI demande lecture de
carte
Retour à ANAL

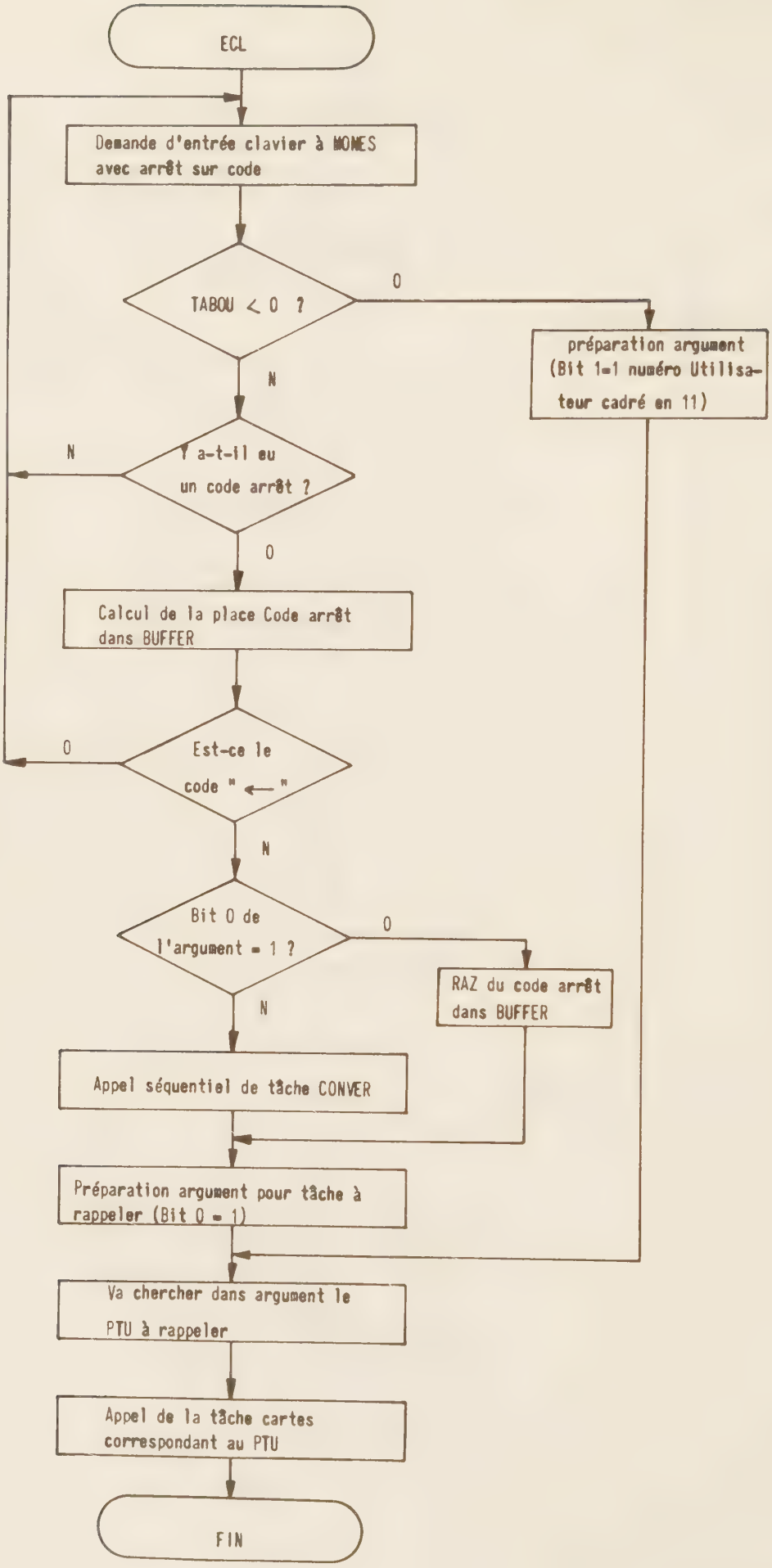

3
Calcul numéro Utilisateur
Est-ce + 8
0
Retour à ANAL
sans rien faire
N
Bit 11 de
TABOU = 1
0
Retour à ANAL
sans rien faire
N
Appel du PTU dans table
attachée à l'appel
Retour à ANAL
2
Envoi caractère NULL
Rangement adresse TTY1 dans
adresse Driver entretien
Retour à ANAL

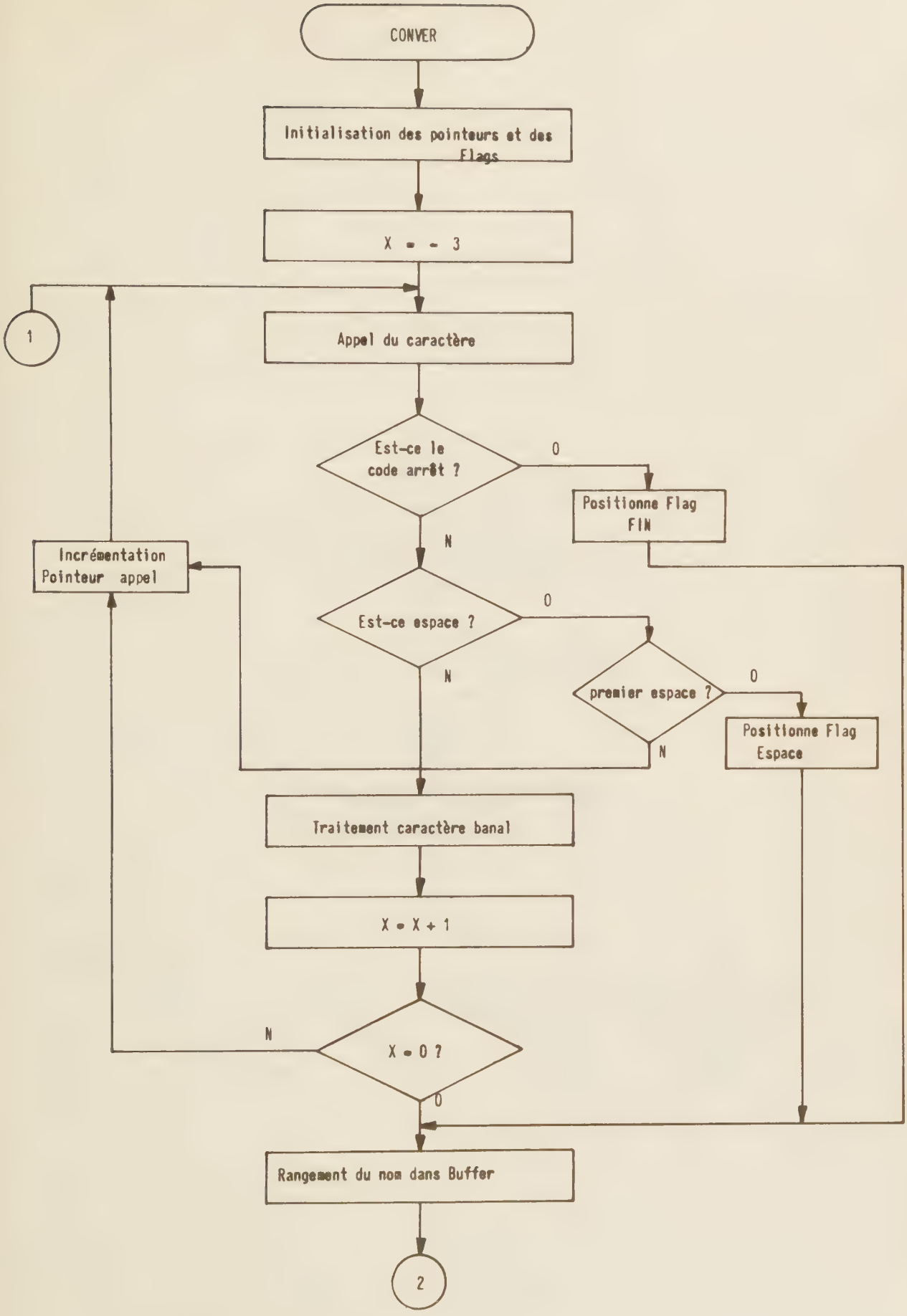

TTY1
Calcul numéro utilisateur
UP occupée
par US COSYNUS
O
N
Appel du PTU en table
4000 → TABOU
A - 040
Retour à MONES
en défaut

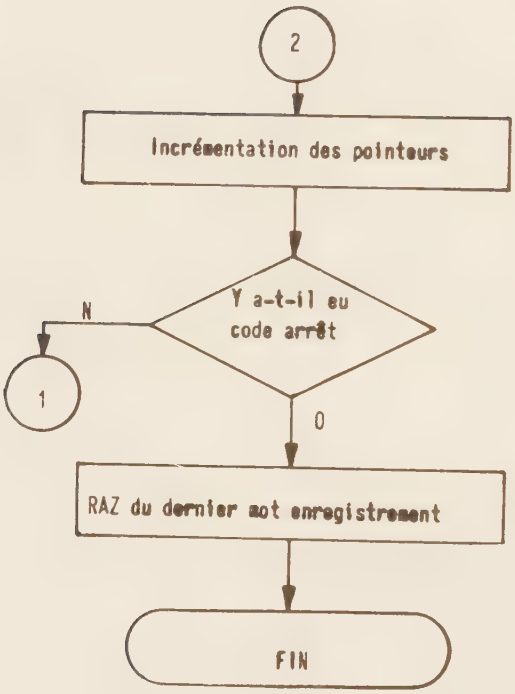

TRCP
Positionne FLAG dans TABOU
Appel à MONES pour entrer 3 caractères
RAZ de TABOU
TABOU < 0 ?
0
N
Conversion 1c/mot en 3 c/mots
Appel tâche ANALYS
FIN

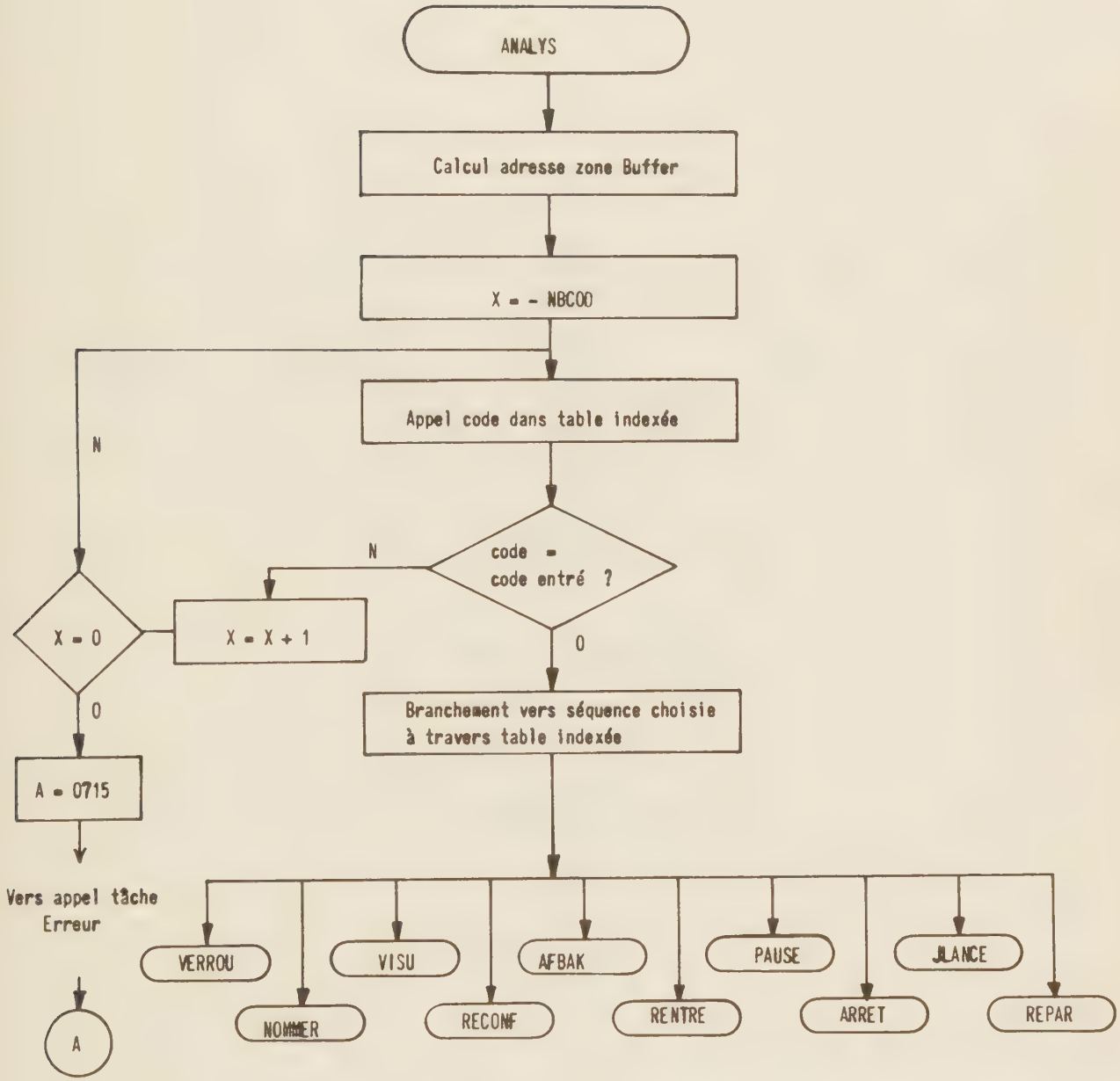

TACLE
Positionne Flag dans TABOU
RAZ de TABOU
Appel à MONES pour entrer
3 caractères
0
TABOU < 0 ?
N
Conversion 1c/mot à 3c/mots
Mot entré =
Mot dépasse ?
N
0
Rangement de PTU TRCP dans
table de PTU
Appel de tâche IMPRI pour
mise TTY OFF
Appel tâche TRCP
RAZ de TABOU
FIN

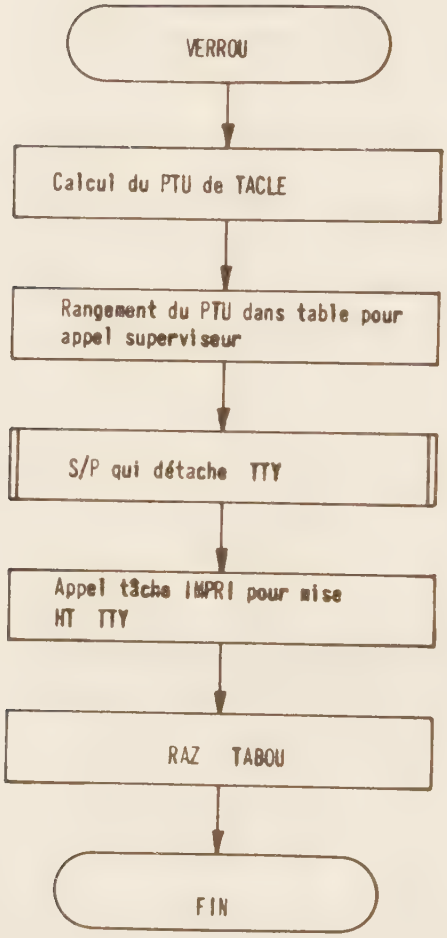

IMPRI
Argument = 0 ?
0
N
Demande à MONES pour mise H.T TTY
Isolation compte de mots
Préparation table échange
Isolation code arrêt
Code arrêt NULL ?
0
N
Code arrêt = + 1 (bidon)
Rangement code arrêt dans table
OS TTY COSYMOS occupée ?
N
0
Demande à MONES pour tuer échange
Demande à MONES impression avec arrêt sur code
FIN

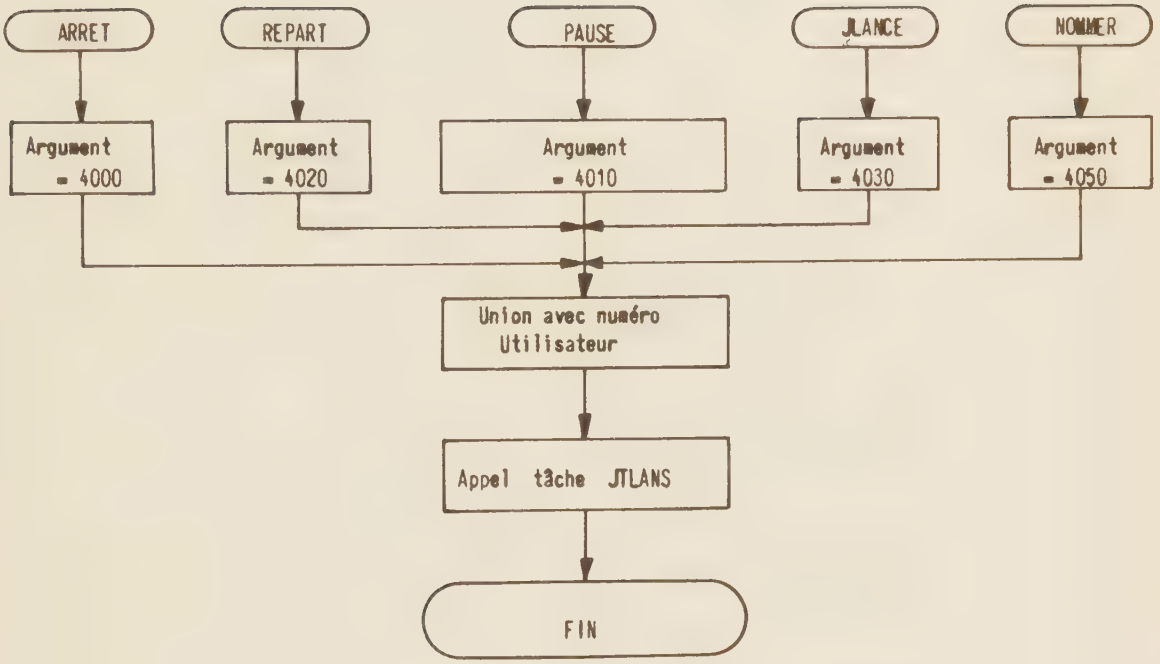

ECL
Demande d'entrée clavier à MOMES avec arrêt sur code
TABOU < 0 ?
O
préparation argument (Bit 1=1 numéro Utilisateur cadré en 11)
N
N
Y a-t-il eu un code arrêt ?
O
Calcul de la place Code arrêt dans BUFFER
O
Est-ce le code " ← "
N
Bit 0 de l'argument = 1 ?
O
RAZ du code arrêt dans BUFFER
N
Appel séquentiel de tâche CONVER
Préparation argument pour tâche à rappeler (Bit 0 = 1)
Va chercher dans argument le PTU à rappeler
Appel de la tâche cartes correspondant au PTU
FIN

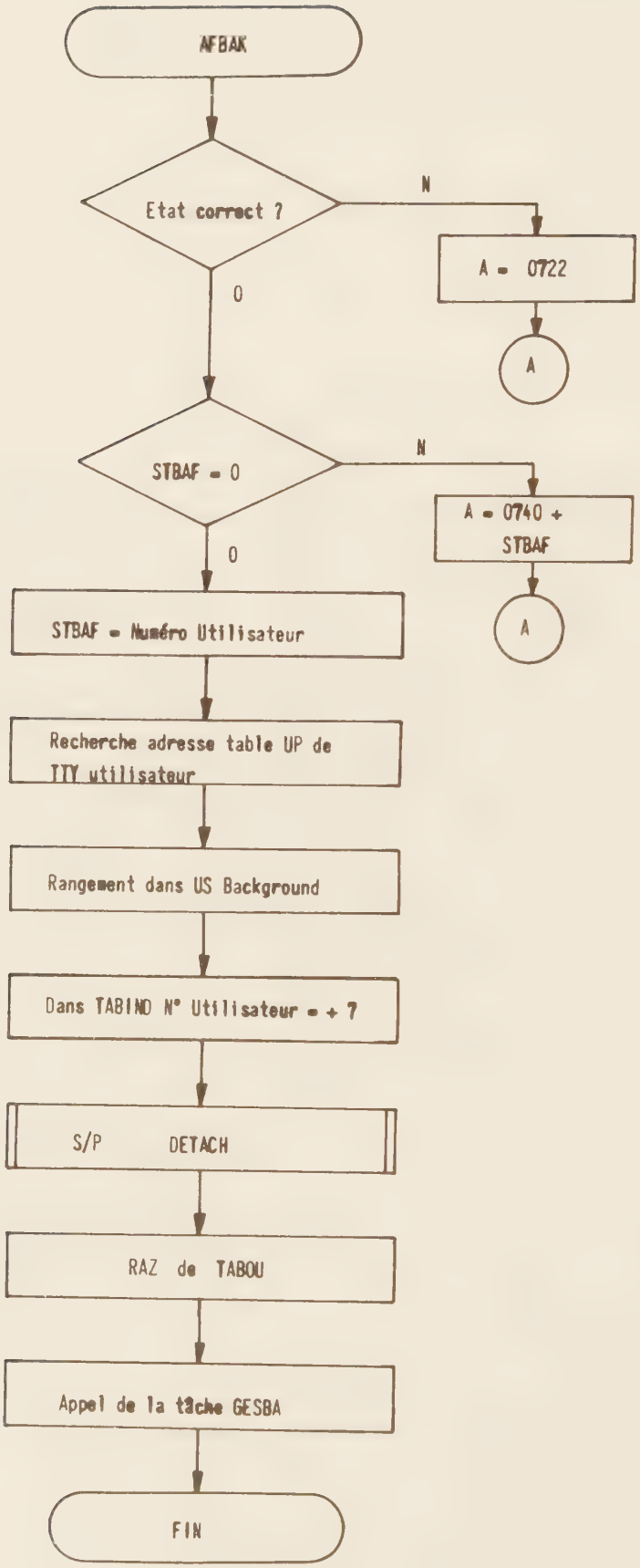

CONVER
Initialisation des pointeurs et des Flags
X = - 3
1
Appel du caractère
Est-ce le code arrêt ?
O
Positionne Flag FIN
N
Incrémentation Pointeur appel
Est-ce espace ?
O
N
premier espace ?
O
Positionne Flag Espace
N
Traitement caractère banal
X = X + 1
X = 0 ?
N
O
Rangement du nom dans Buffer
2

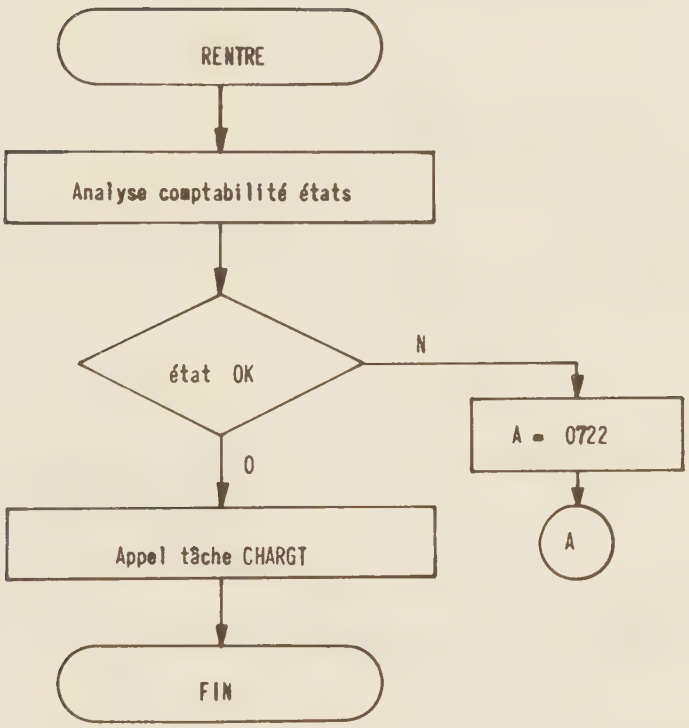

2
Incrémentation des pointeurs
Y a-t-il eu
code arrêt
N
1
O
RAZ du dernier mot enregistrement
FIN

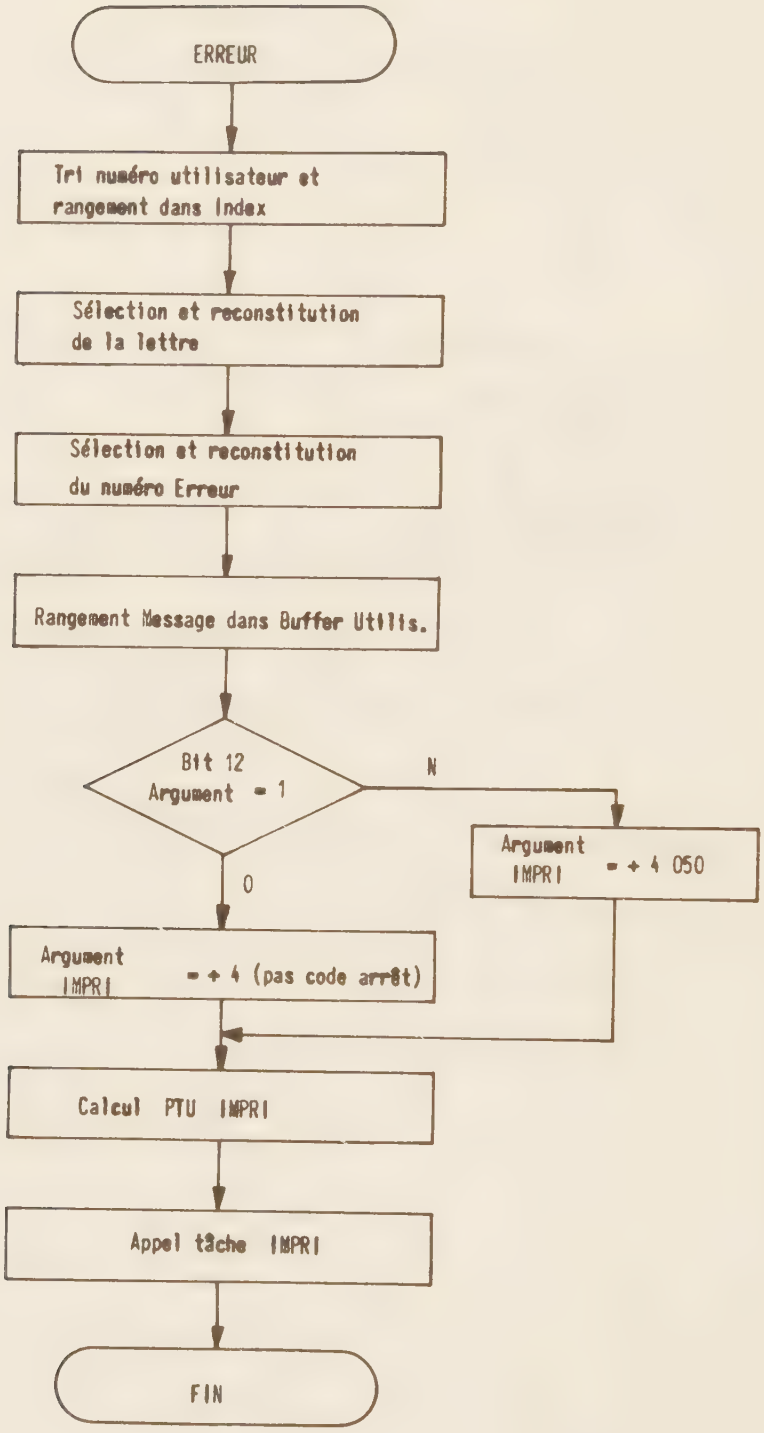

ANALYS
Calcul adresse zone Buffer
X = - NBCOD
Appel code dans table indexée
N
code =
code entré ?
N
X = 0
X = X + 1
0
Branchement vers séquence choisie
à travers table indexée
0
A = 0715
Vers appel tâche
Erreur
A
VERROU
NOMMER
VISU
RECONF
AFBAK
RENTRE
PAUSE
ARRET
JLANCE
REPAR

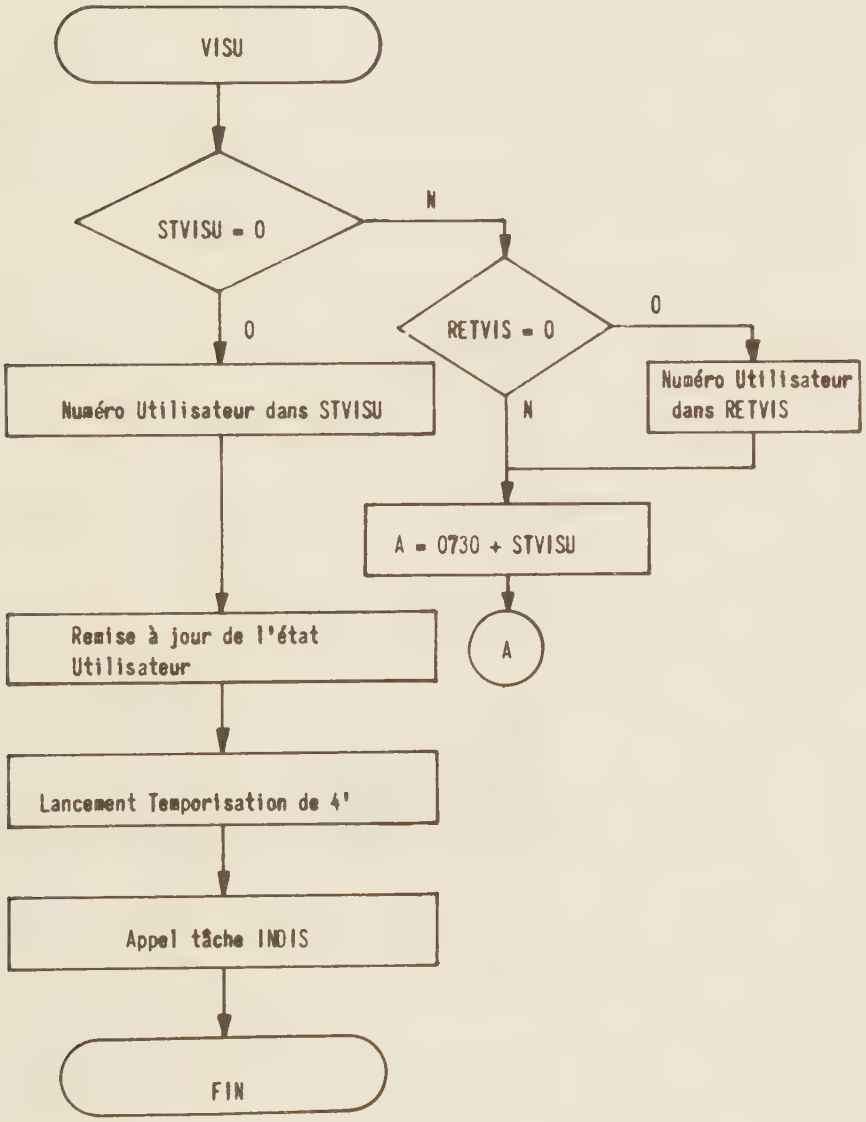

VERROU
Calcul du PTU de TACLE
Rangement du PTU dans table pour appel superviseur
S/P qui détache TTY
Appel tâche IMPRI pour mise HT TTY
RAZ TABOU
FIN

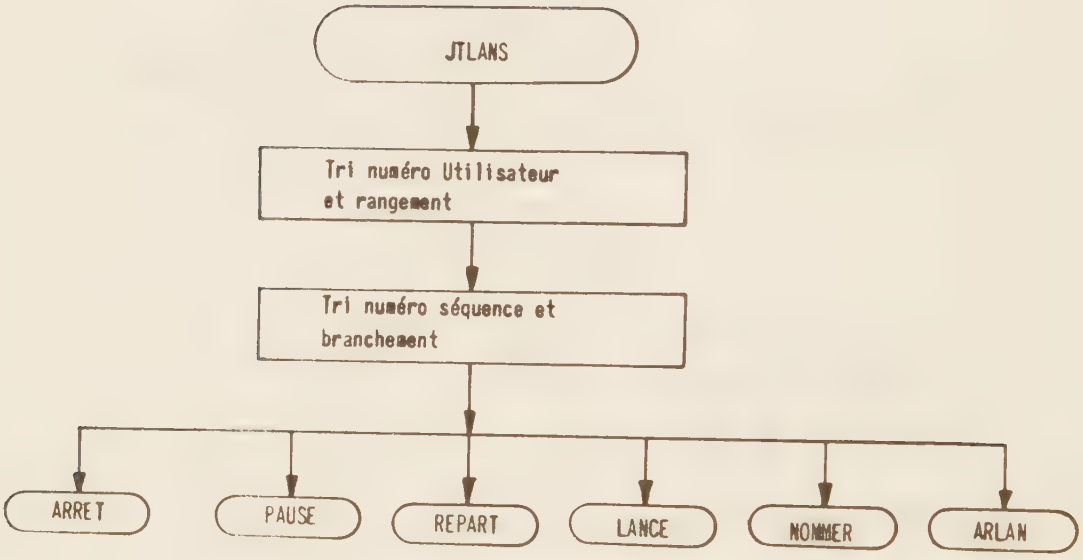

ARRET
REPART
PAUSE
JLANCE
NOMMER
Argument = 4000
Argument = 4020
Argument = 4010
Argument = 4030
Argument = 4050
Union avec numéro Utilisateur
Appel tâche JTLANS
FIN

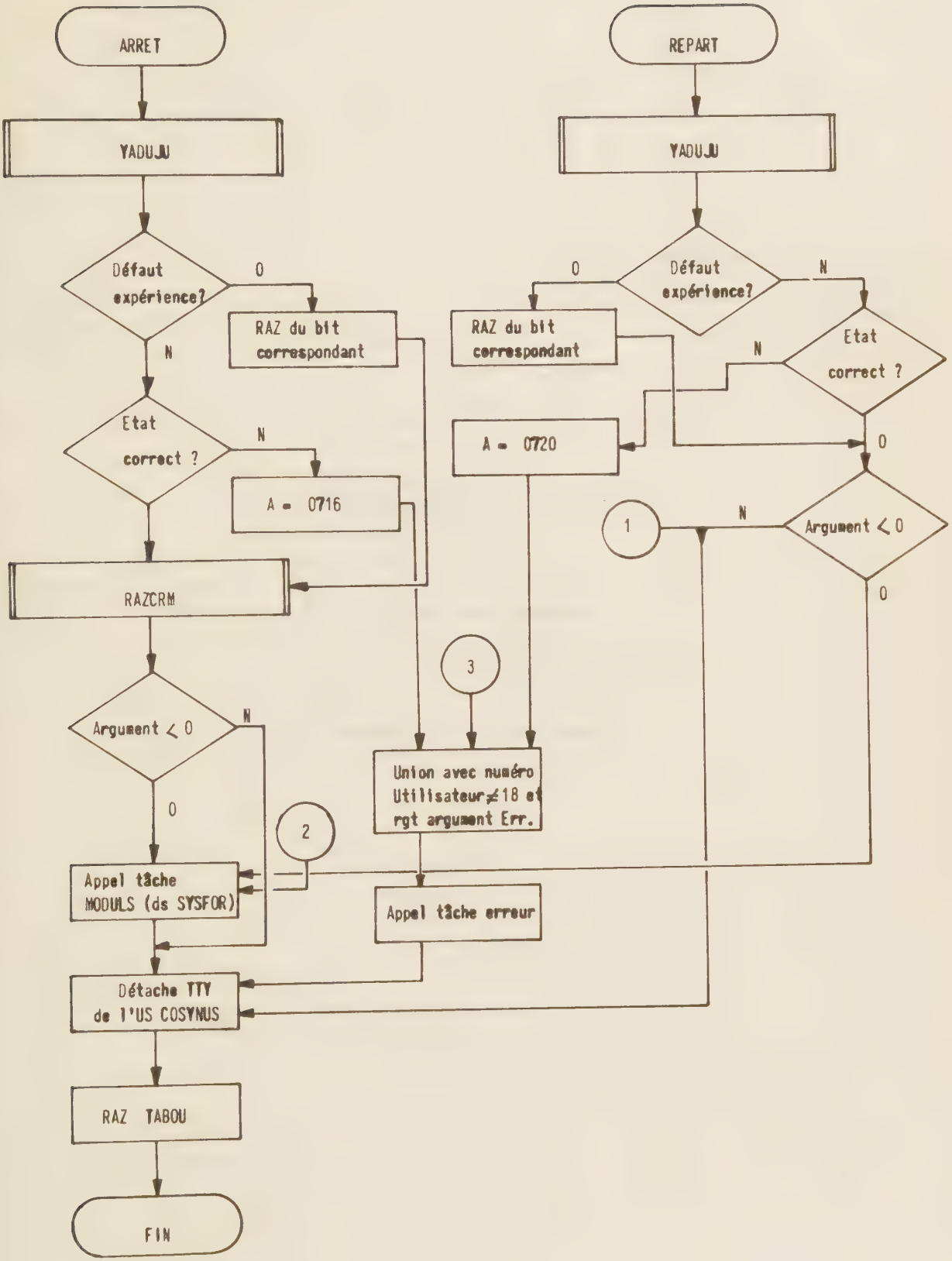

AFBAK
Etat correct ?
N
A = 0722
A
O
STBAF = 0
N
A = 0740 +
STBAF
A
O
STBAF = Numéro Utilisateur
Recherche adresse table UP de
TTY utilisateur
Rangement dans US Background
Dans TABIND N° Utilisateur = + ?
S/P DETACH
RAZ de TABOU
Appel de la tâche GESBA
FIN

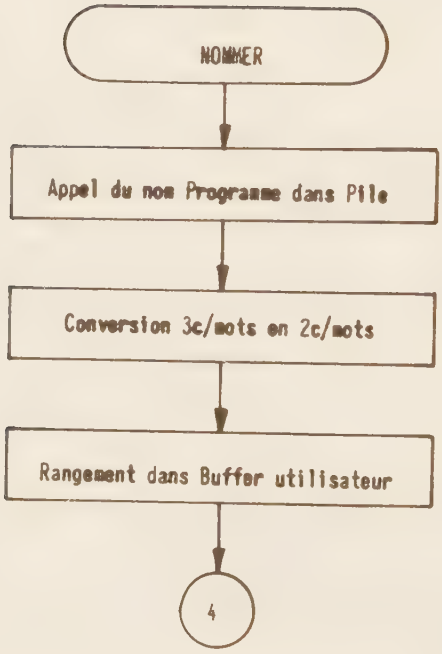

RENTRE
Analyse comptabilité états
état OK
N
A = 0722
A
0
Appel tâche CHARGT
FIN

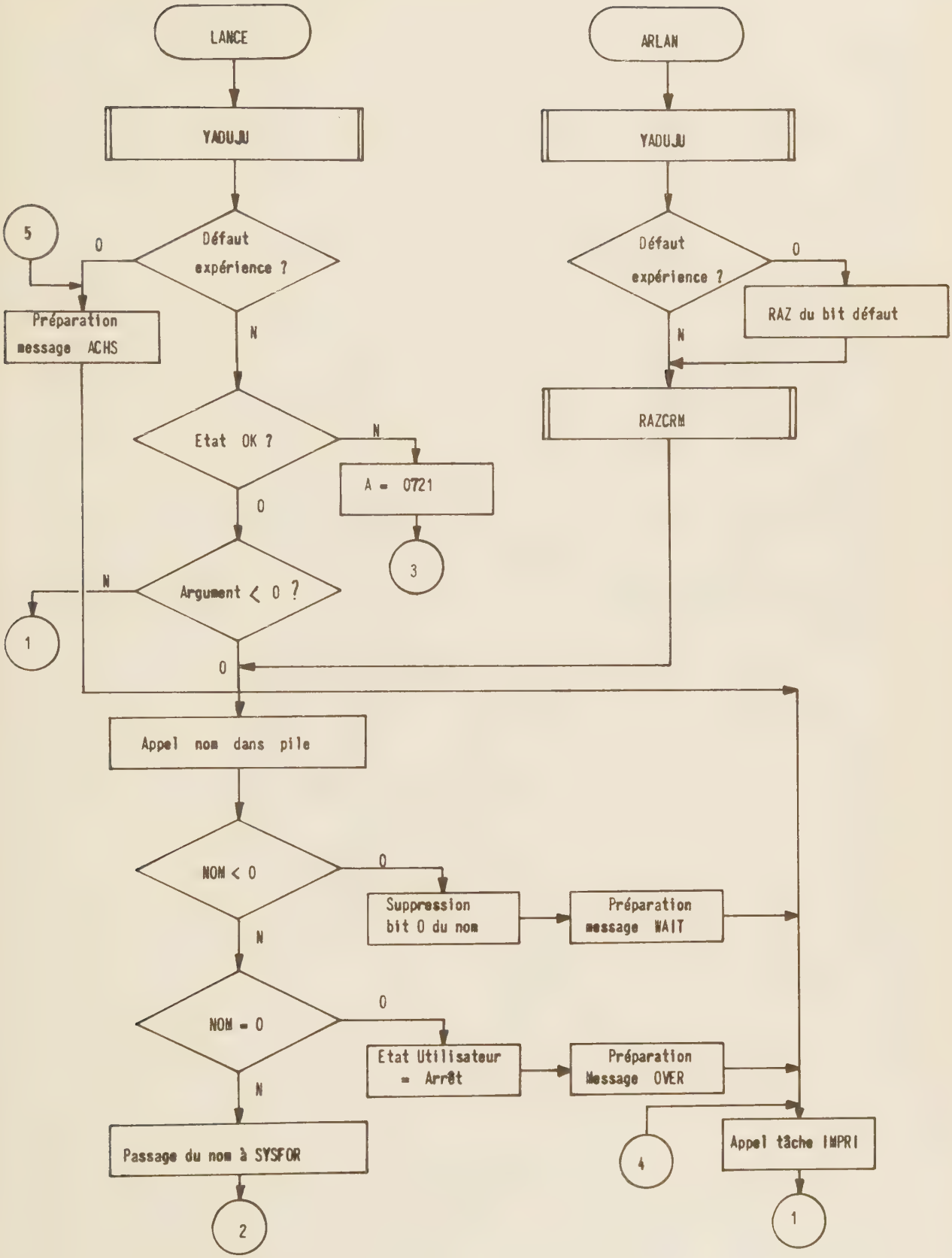

ERREUR
Tri numéro utilisateur et rangement dans Index
Sélection et reconstitution de la lettre
Sélection et reconstitution du numéro Erreur
Rangement Message dans Buffer Utilis.
Bit 12 Argument = 1
N
0
Argument IMPRI = + 4 050
Argument IMPRI = + 4 (pas code arrêt)
Calcul PTU IMPRI
Appel tâche IMPRI
FIN

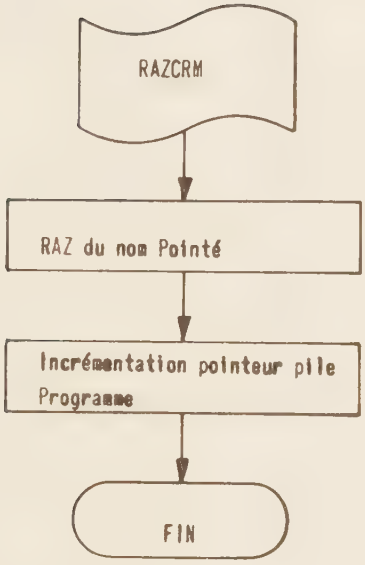

VISU
STVISU = 0
N
0
RETVIS = 0
0
N
Numéro Utilisateur dans STVISU
Numéro Utilisateur dans RETVIS
A = 0730 + STVISU
A
Remise à jour de l'état Utilisateur
Lancement Temporisation de 4'
Appel tâche INDIS
FIN

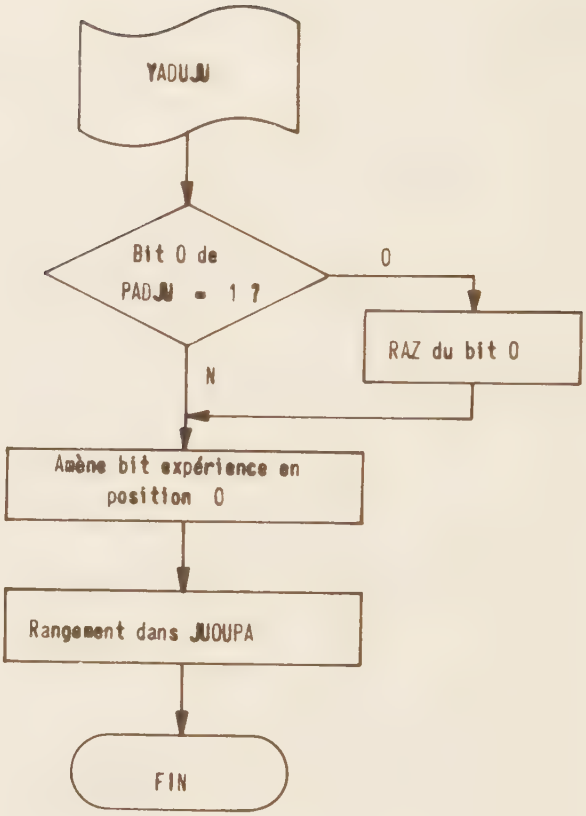

JTLANS
Tri numéro Utilisateur et rangement
Tri numéro séquence et branchement
ARRET
PAUSE
REPART
LANCE
NOMMER
ARLAN

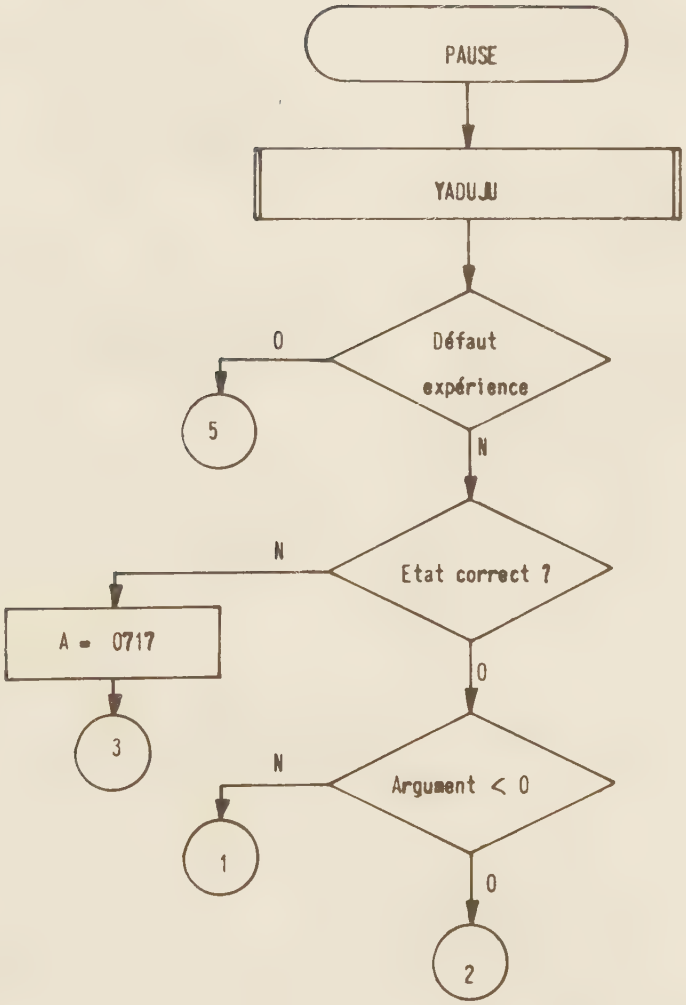

ARRET
YADUJU
Défaut expérience?
O
RAZ du bit correspondant
N
Etat correct ?
N
A = 0716
RAZCRM
Argument < 0
N
O
2
Appel tâche MODULS (ds SYSFOR)
Détache TTY de l'US COSYNUS
RAZ TABOU
FIN
REPART
YADUJU
Défaut expérience?
O
RAZ du bit correspondant
N
Etat correct ?
N
A = 0720
O
1
N
Argument < 0
O
3
Union avec numéro Utilisateur≠18 et rgt argument Err.
Appel tâche erreur

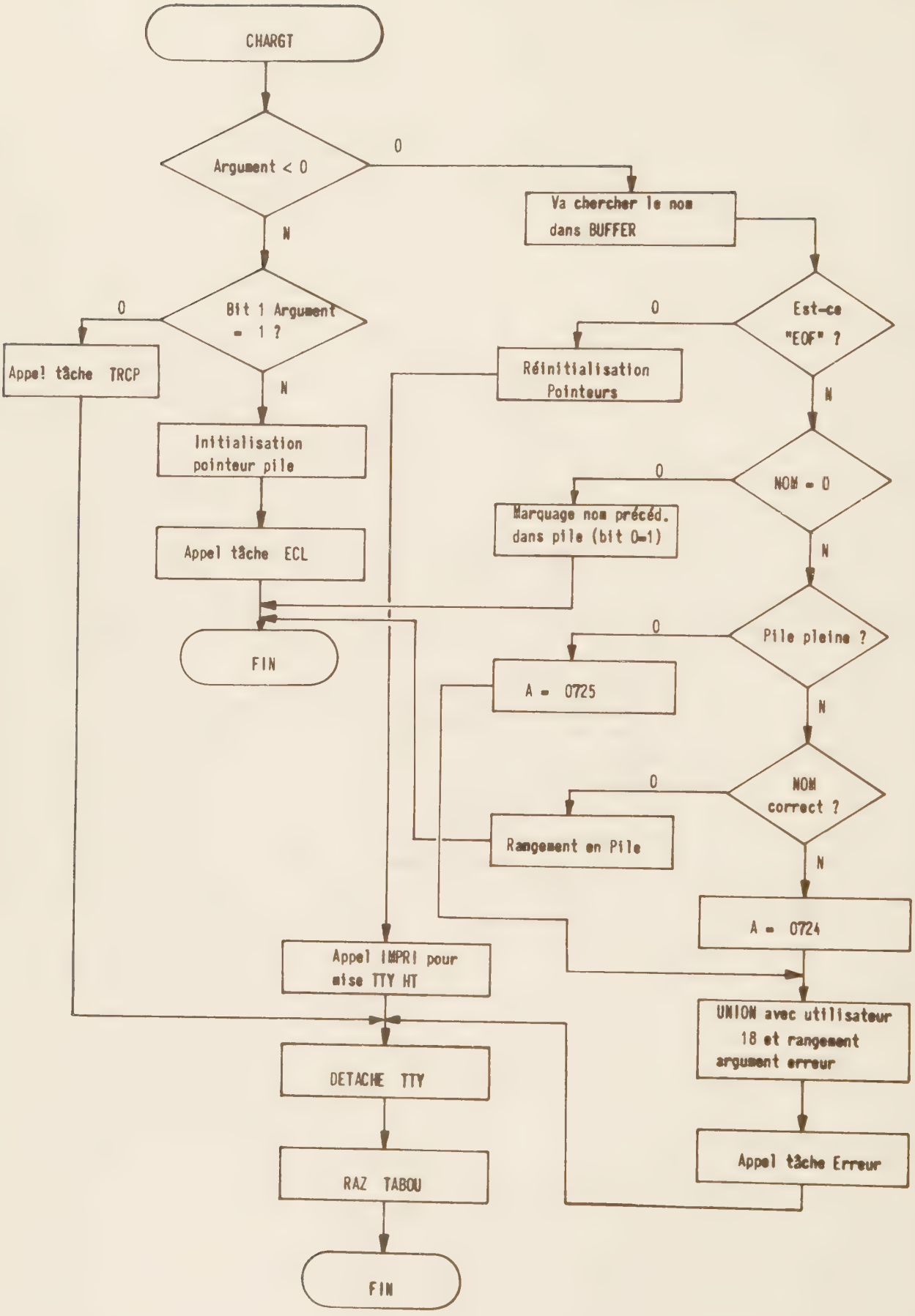

NOMMER
Appel du nom Programme dans Pile
Conversion 3c/mots en 2c/mots
Rangement dans Buffer utilisateur
4

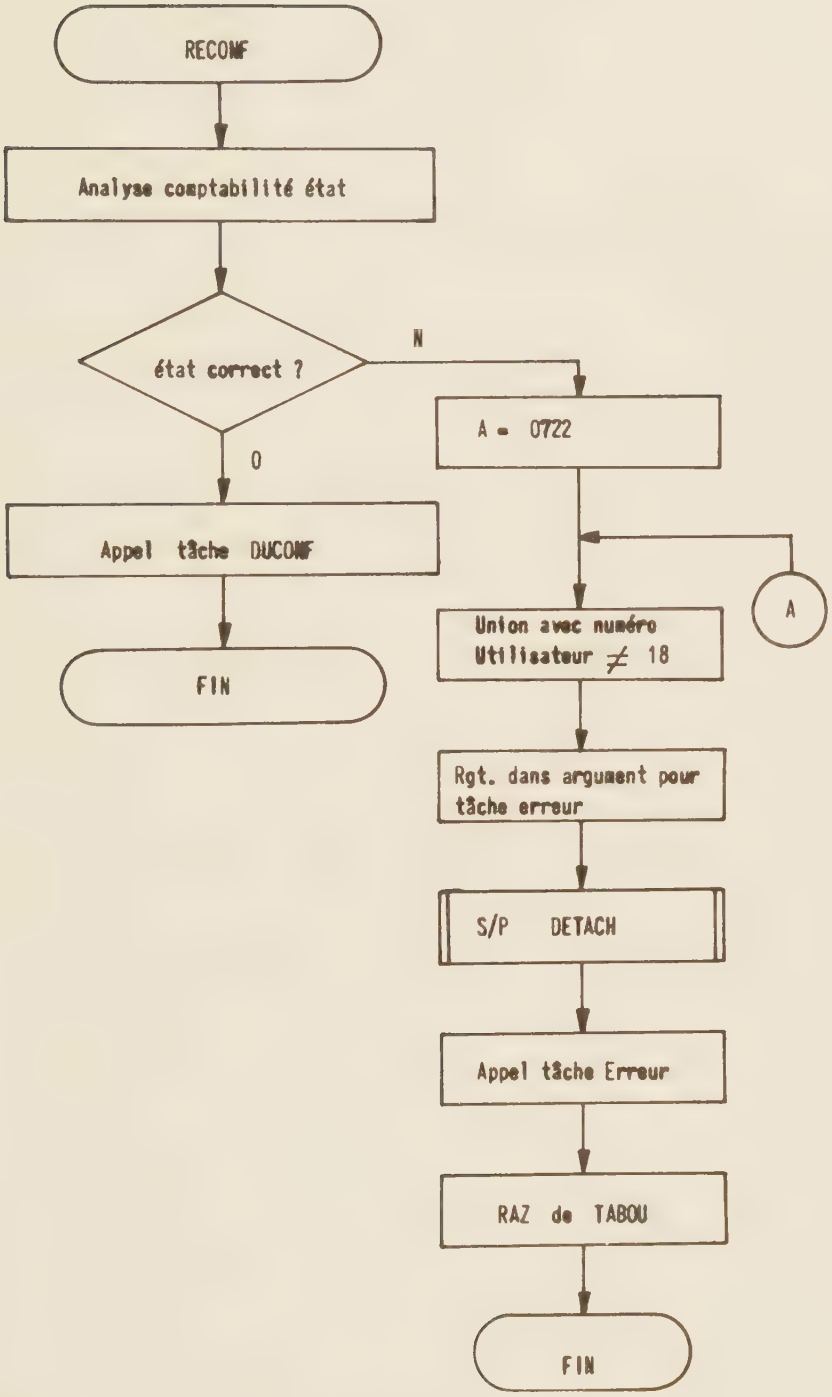

LANCE
YADUJU
5
Défaut expérience ?
O
N
Préparation message ACHS
Etat OK ?
N
A = 0721
3
O
Argument < 0 ?
N
1
O
Appel nom dans pile
NOM < 0
O
Suppression bit 0 du nom
Préparation message WAIT
N
NOM = 0
O
Etat Utilisateur = Arrêt
Préparation Message OVER
N
Passage du nom à SYSFOR
2
ARLAN
YADUJU
Défaut expérience ?
O
RAZ du bit défaut
N
RAZCRM
4
Appel tâche IMPRI
1


RAZCRM
RAZ du nom Pointé
Incrémentation pointeur pile Programme
FIN


YADUJU
Bit 0 de PADJU = 1 ?
0
RAZ du bit 0
N
Amène bit expérience en position 0
Rangement dans JUOUPA
FIN


PAUSE
YADUJU
Défaut expérience
O
5
N
Etat correct ?
N
A = 0717
3
O
Argument < 0
N
1
O
2

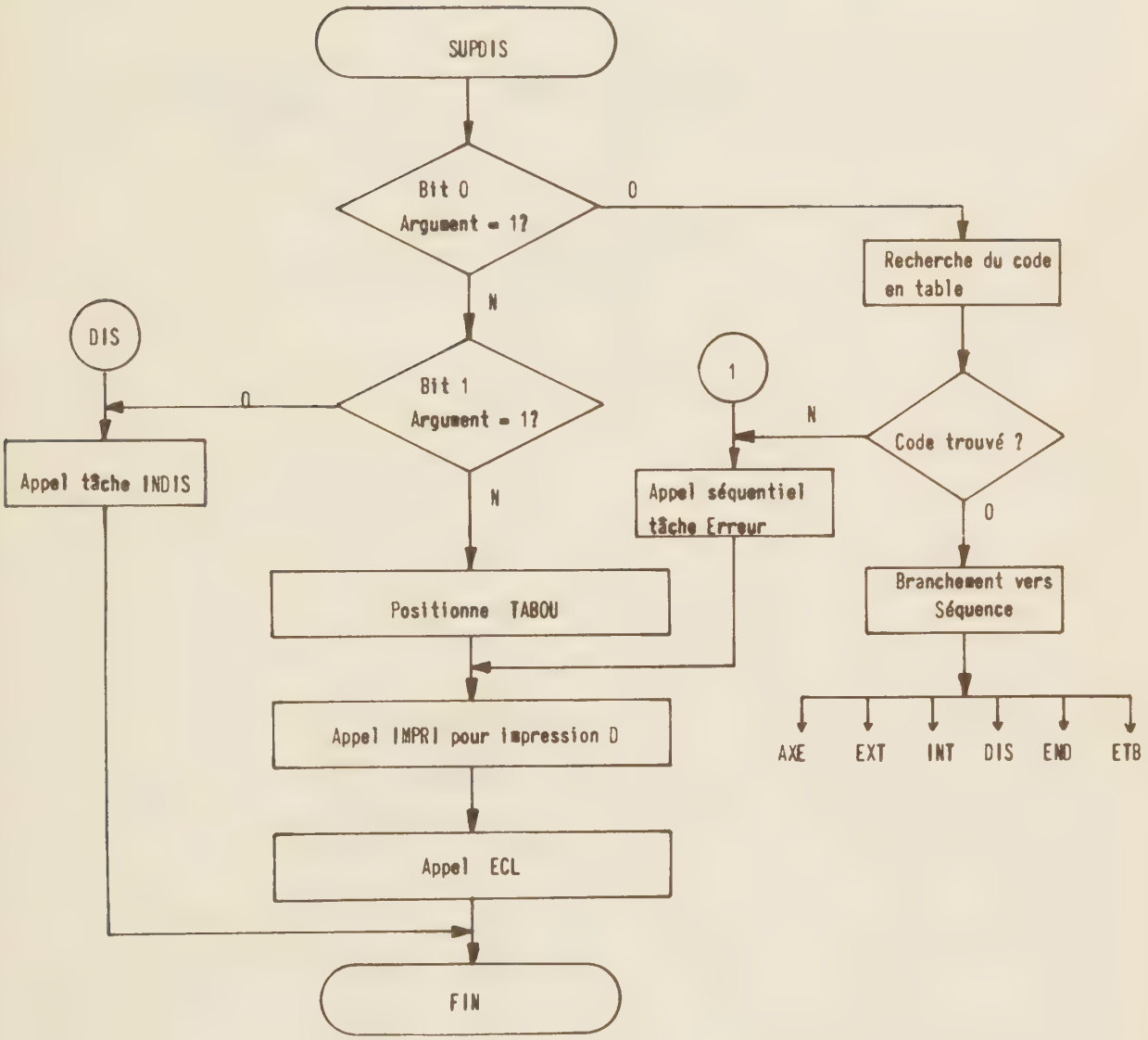

CHARGT
Argument < 0
0
N
Va chercher le nom dans BUFFER
Bit 1 Argument = 1 ?
0
N
Appel tâche TRCP
Initialisation pointeur pile
Appel tâche ECL
FIN
Est-ce "EOF" ?
0
N
Réinitialisation Pointeurs
NOM = 0
0
N
Marquage nom précéd. dans pile (bit 0=1)
Pile pleine ?
0
N
A = 0725
NOM correct ?
0
N
Rangement en Pile
A = 0724
Appel IMPRI pour mise TTY HT
UNION avec utilisateur 18 et rangement argument erreur
Appel tâche Erreur
DETACHE TTY
RAZ TABOU
FIN

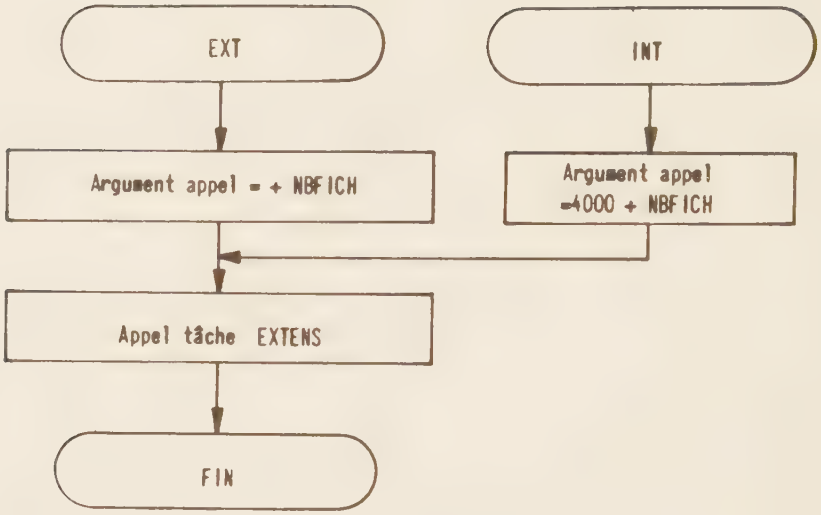

RECONF
Analyse comptabilité état
état correct ?
N
0
A = 0722
Appel tâche DUCONF
FIN
A
Union avec numéro
Utilisateur ≠ 18
Rgt. dans argument pour
tâche erreur
S/P DETACH
Appel tâche Erreur
RAZ de TABOU
FIN

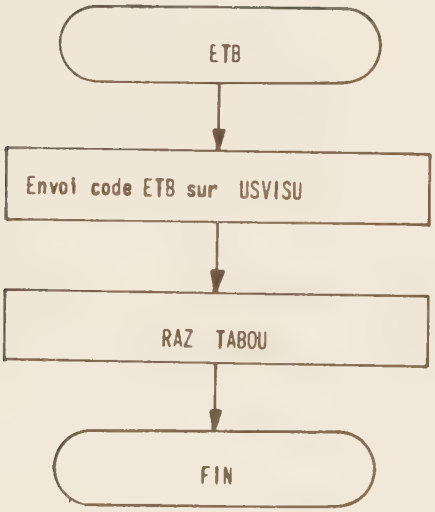
ANNEXE II

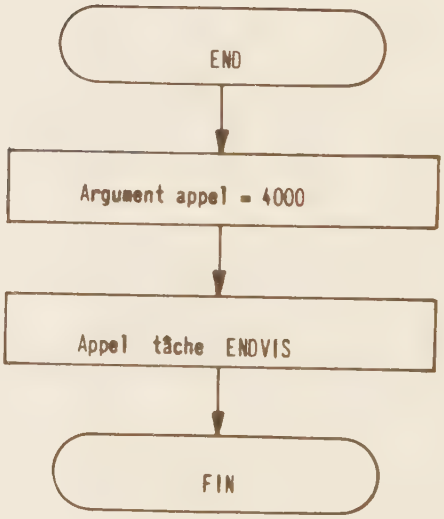
ORGANIGRAMMES DE PRINCIPE

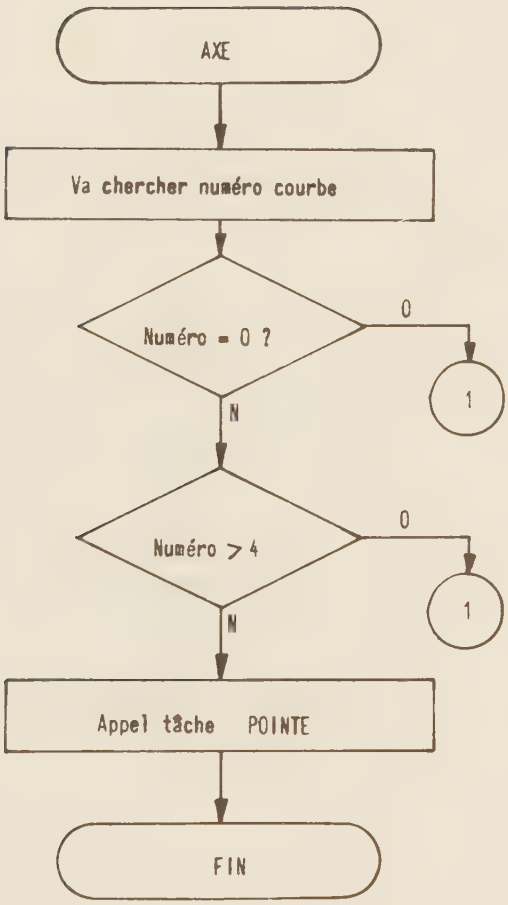
DES TACHES GERANT LE DISPLAY

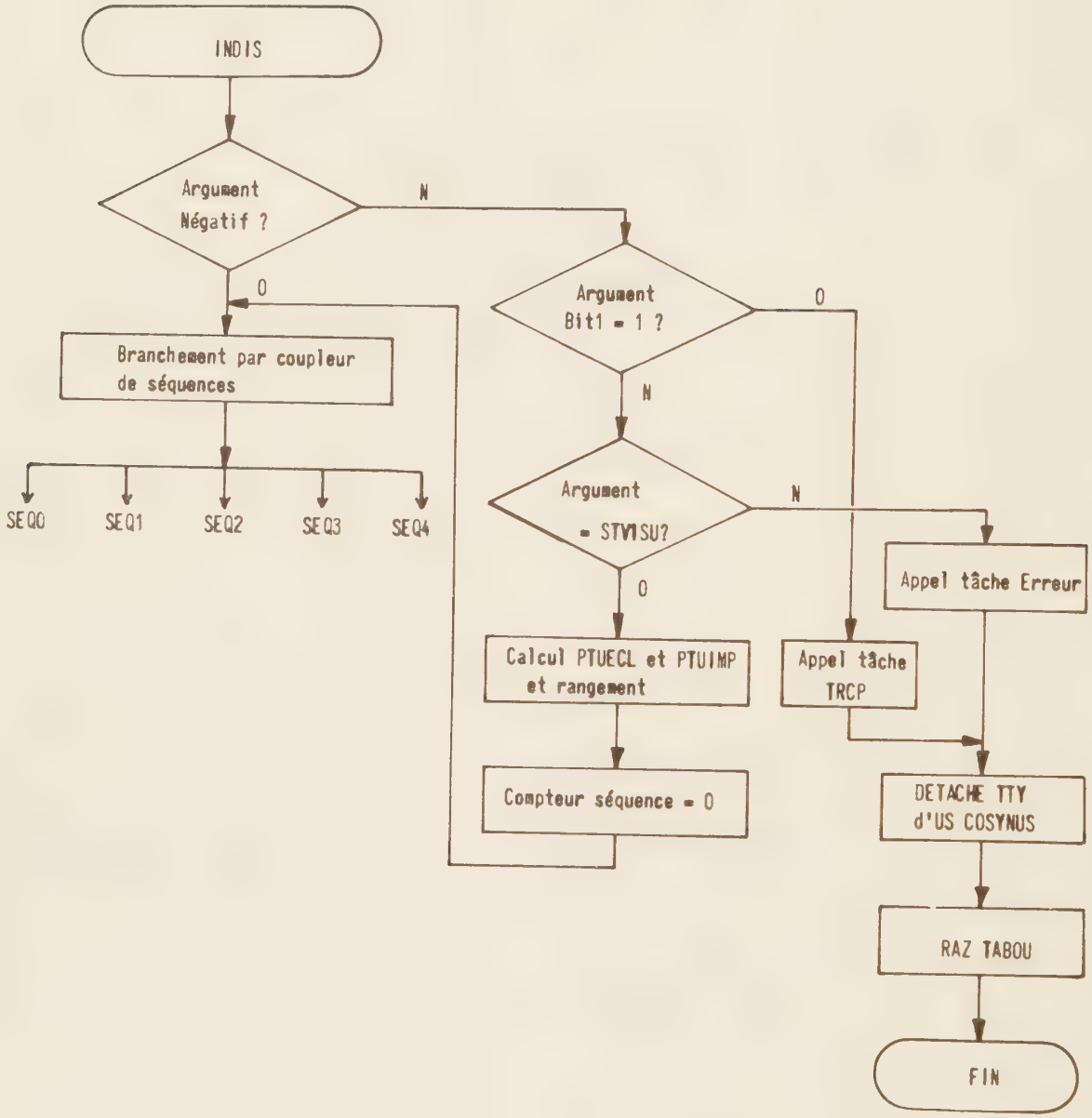

SUPDIS
Bit 0
Argument = 1?
0
N
Recherche du code
en table
DIS
Bit 1
Argument = 1?
0
N
1
N
Code trouvé ?
0
Appel tâche INDIS
Appel séquentiel
tâche Erreur
Positionne TABOU
Branchement vers
Séquence
Appel IMPRI pour impression D
AXE
EXT
INT
DIS
END
ETB
Appel ECL
FIN

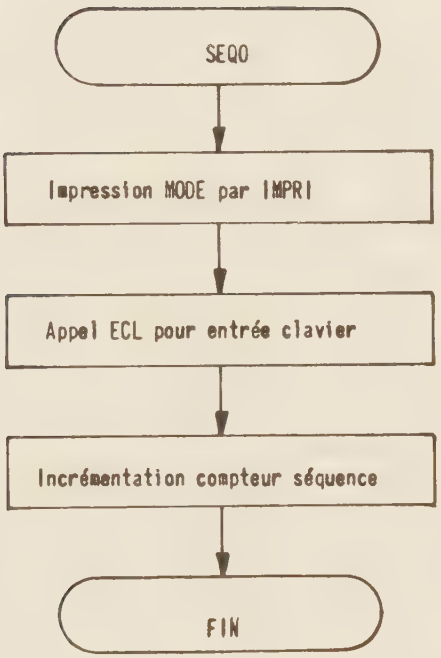

EXT
INT
Argument appel = + NBFICH
Argument appel
=4000 + NBFICH
Appel tâche EXTENS
FIN

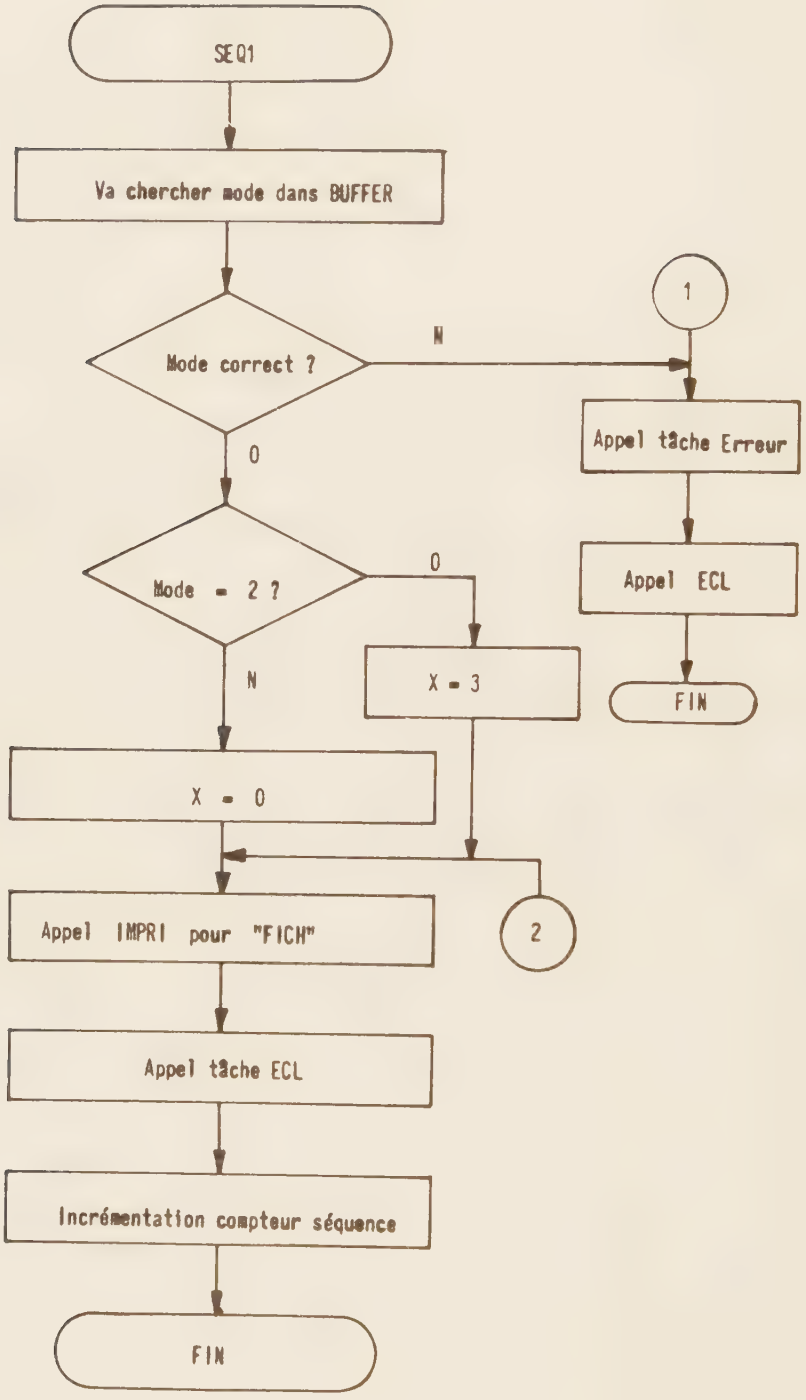

ETB
Envoi code ETB sur USVISU
RAZ TABOU
FIN

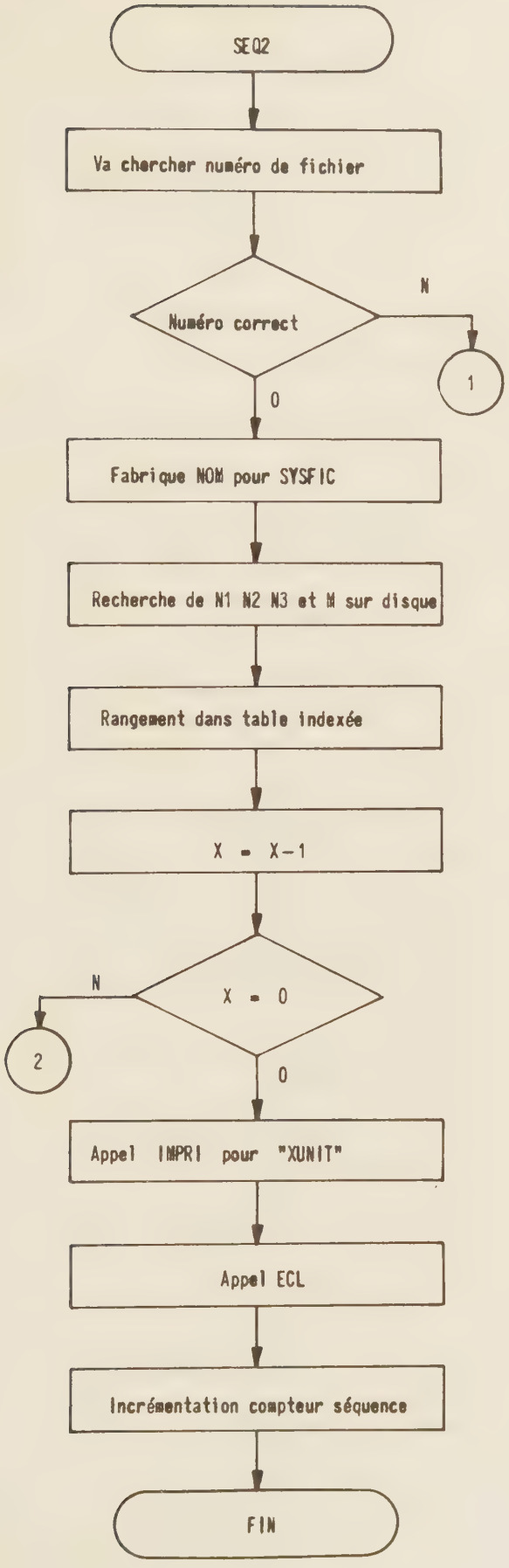

END
Argument appel = 4000
Appel tâche ENDVIS
FIN

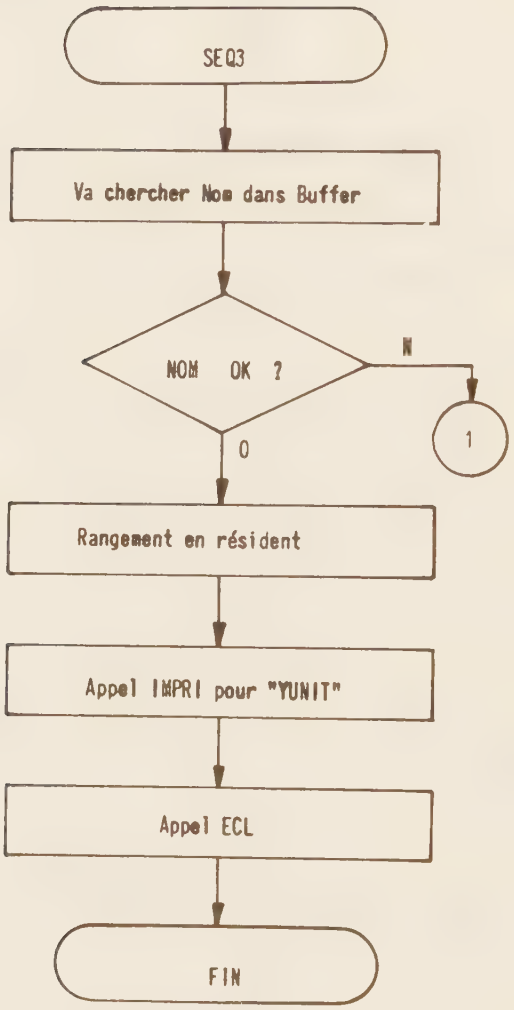

AXE
Va chercher numéro courbe
Numéro = 0 ?
0
1
N
Numéro > 4
0
1
N
Appel tâche POINTE
FIN

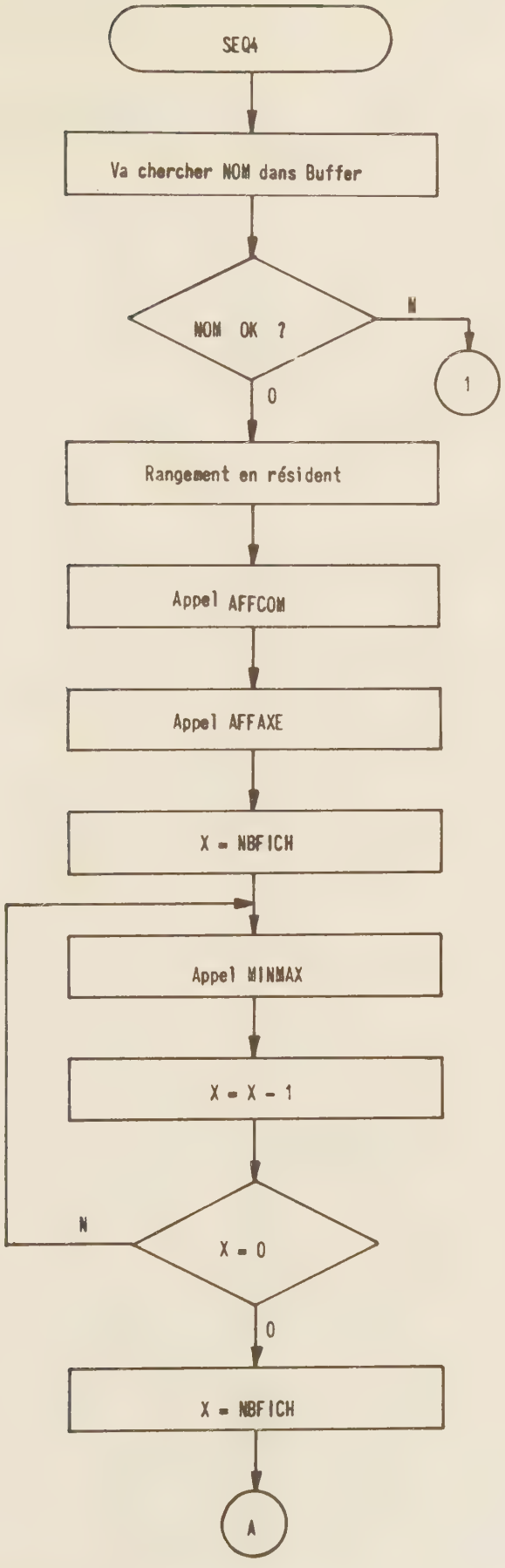

INDIS
Argument Négatif ?
N
O
Branchement par coupleur de séquences
SEQ0
SEQ1
SEQ2
SEQ3
SEQ4
Argument Bit1 = 1 ?
O
N
Argument = STVISU?
N
O
Appel tâche Erreur
Calcul PTUECL et PTUIMP et rangement
Appel tâche TRCP
Compteur séquence = 0
DETACHE TTY d'US COSYNUS
RAZ TABOU
FIN

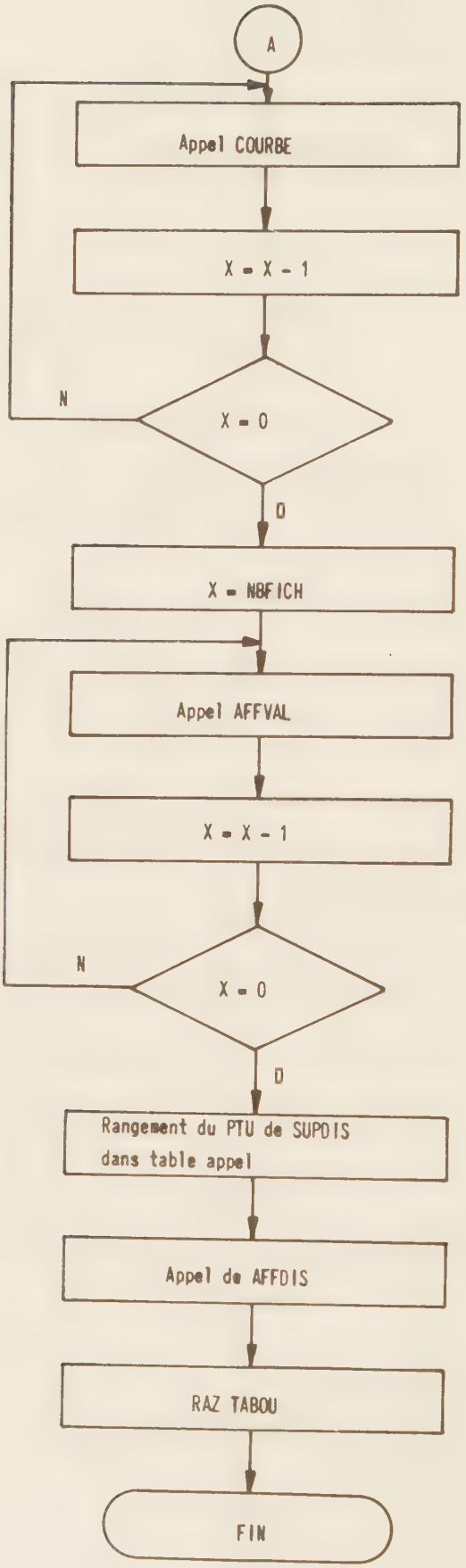

SEQO
Impression MODE par IMPRI
Appel ECL pour entrée clavier
Incrémentation compteur séquence
FIN


SEQ1
Va chercher mode dans BUFFER
Mode correct ?
N
1
Appel tâche Erreur
Appel ECL
FIN
O
Mode = 2 ?
O
X = 3
N
X = 0
Appel IMPRI pour "FICH"
2
Appel tâche ECL
Incrémentation compteur séquence
FIN


SEQ2
Va chercher numéro de fichier
Numéro correct
N
1
O
Fabrique NOM pour SYSFIC
Recherche de N1 N2 N3 et M sur disque
Rangement dans table indexée
X = X-1
X = 0
N
2
O
Appel IMPRI pour "XUNIT"
Appel ECL
Incrémentation compteur séquence
FIN


SEQ3
Va chercher Nom dans Buffer
NOM OK ?
N
1
O
Rangement en résident
Appel IMPRI pour "YUNIT"
Appel ECL
FIN

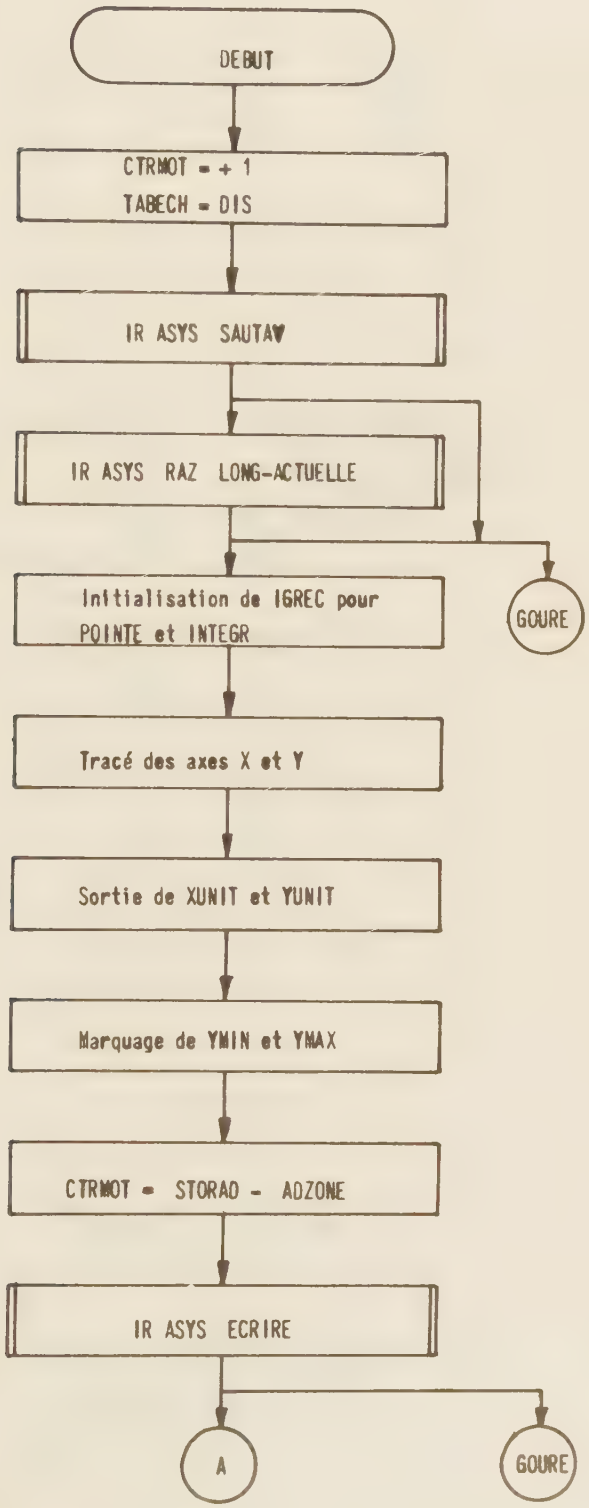

SEQ4
Va chercher NOM dans Buffer
NOM OK ?
N
1
O
Rangement en résident
Appel AFFCOM
Appel AFFAXE
X = NBFICH
Appel MINMAX
X = X - 1
N
X = 0
O
X = NBFICH
A

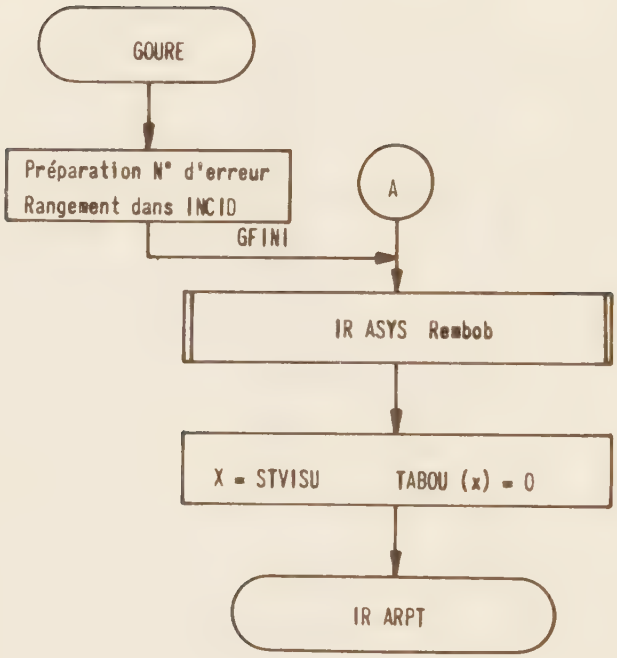

A
Appel COURBE
X = X - 1
X = 0
N
O
X = NBFICH
Appel AFFVAL
X = X - 1
X = 0
N
O
Rangement du PTU de SUPDIS
dans table appel
Appel de AFFDIS
RAZ TABOU
FIN

B - DESCRIPTION DETAILLEE

I - ORGANIGRAMME DE PRINCIPE

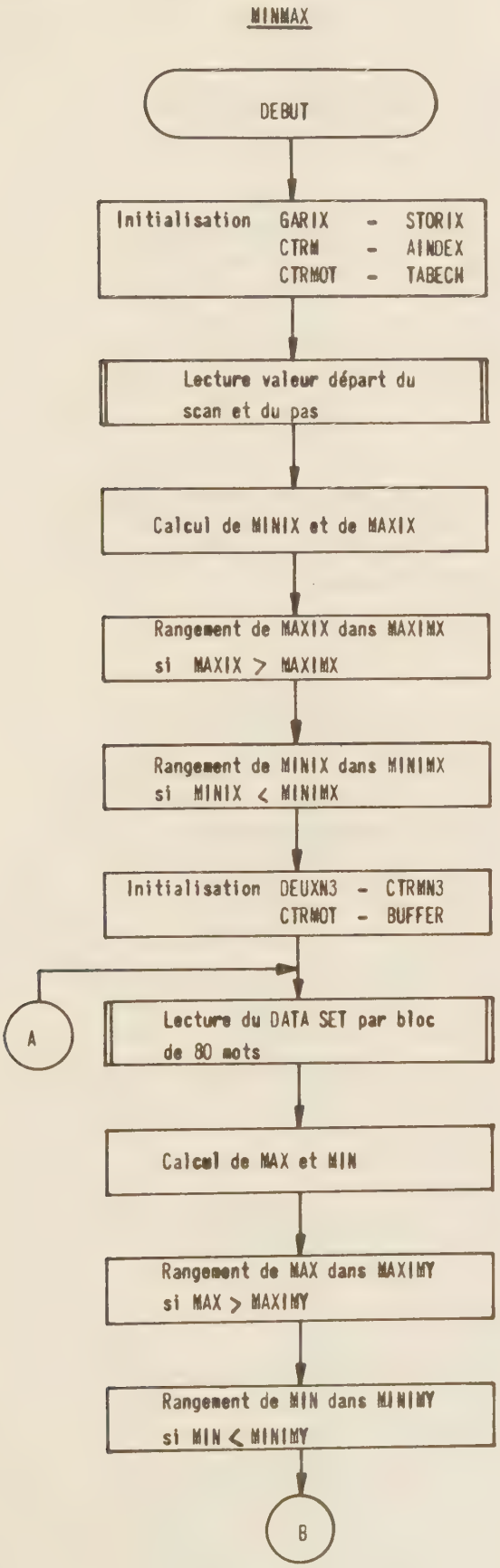
AFFAXE

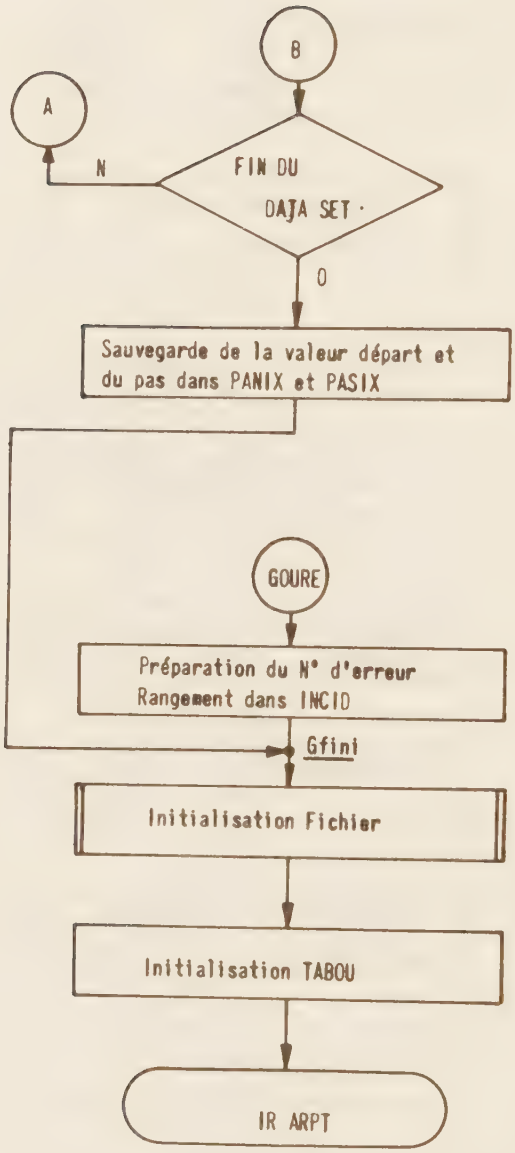



GOURE
Préparation N° d'erreur
Rangement dans INCID
GFINI
A
IR ASYS Rembob
X = STVISU
TABOU (x) = 0
IR ARPT

B - DESCRIPTION DETAILLEE

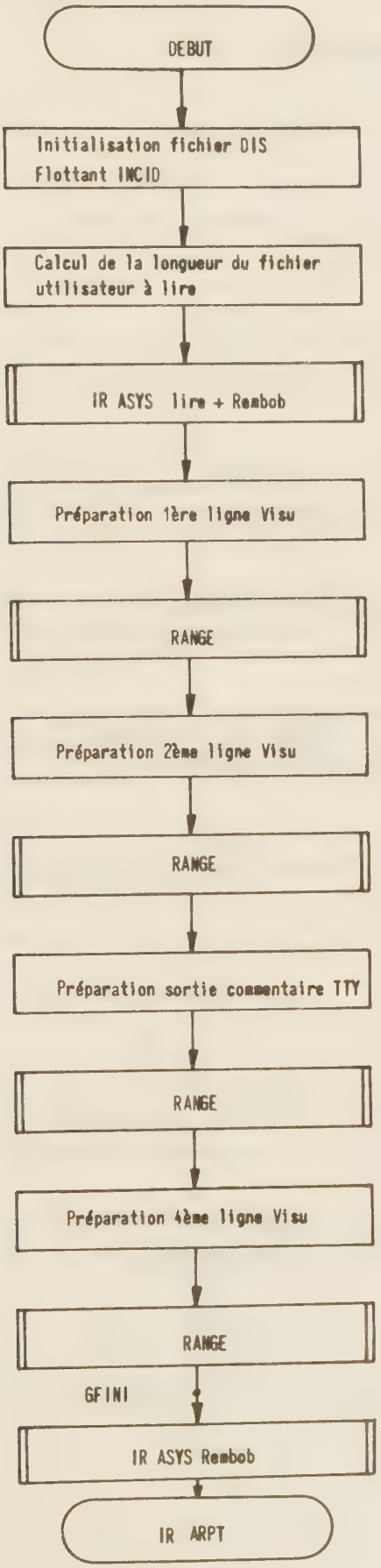
I - ORGANIGRAMME DE PRINCIPE




B
A
N
FIN DU
DAJA SET ·
O
Sauvegarde de la valeur départ et
du pas dans PANIX et PASIX
GOURE
Préparation du N° d'erreur
Rangement dans INCID
Gfini
Initialisation Fichier
Initialisation TABOU
IR ARPT

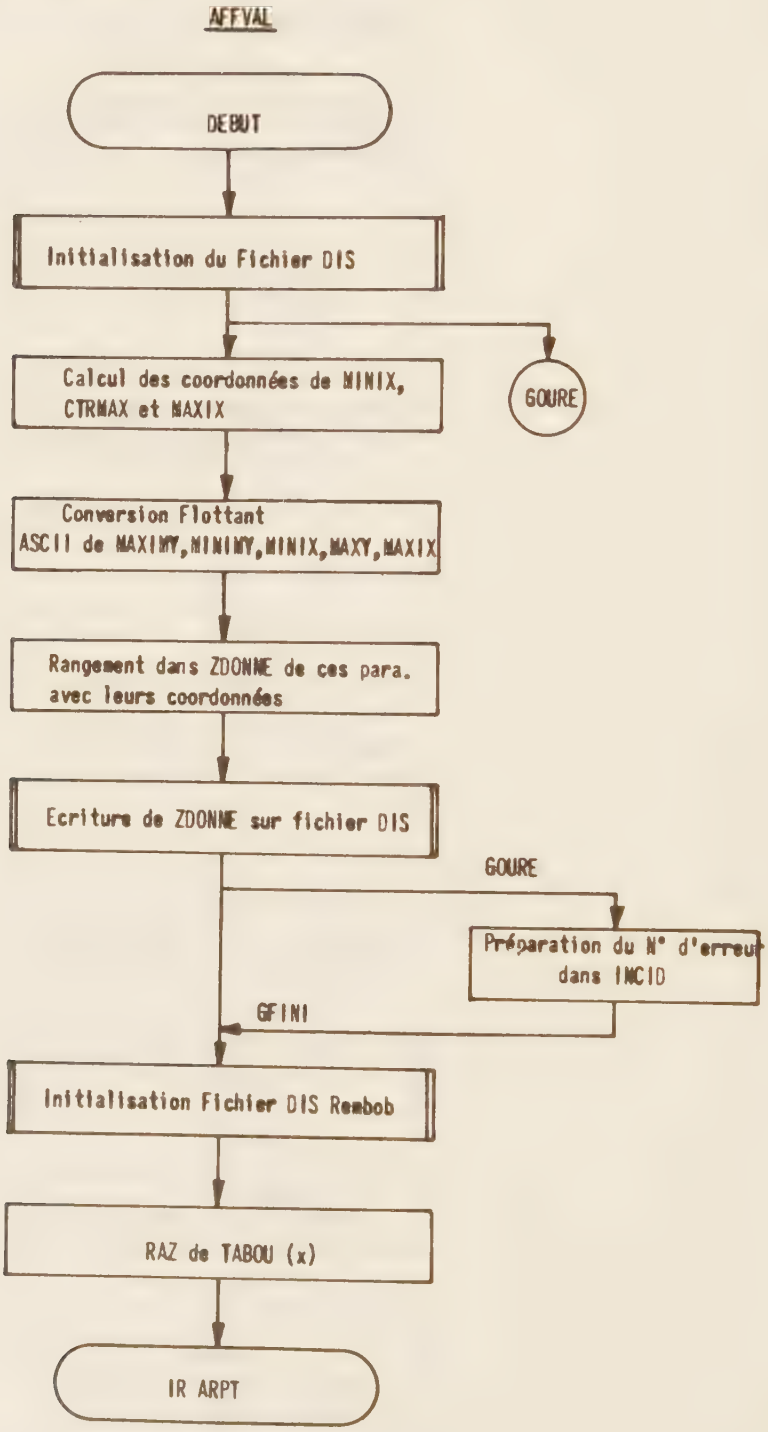
B - DESCRIPTION DETAILLEE

I - ORGANIGRAMME DE PRINCIPE AFFCOM

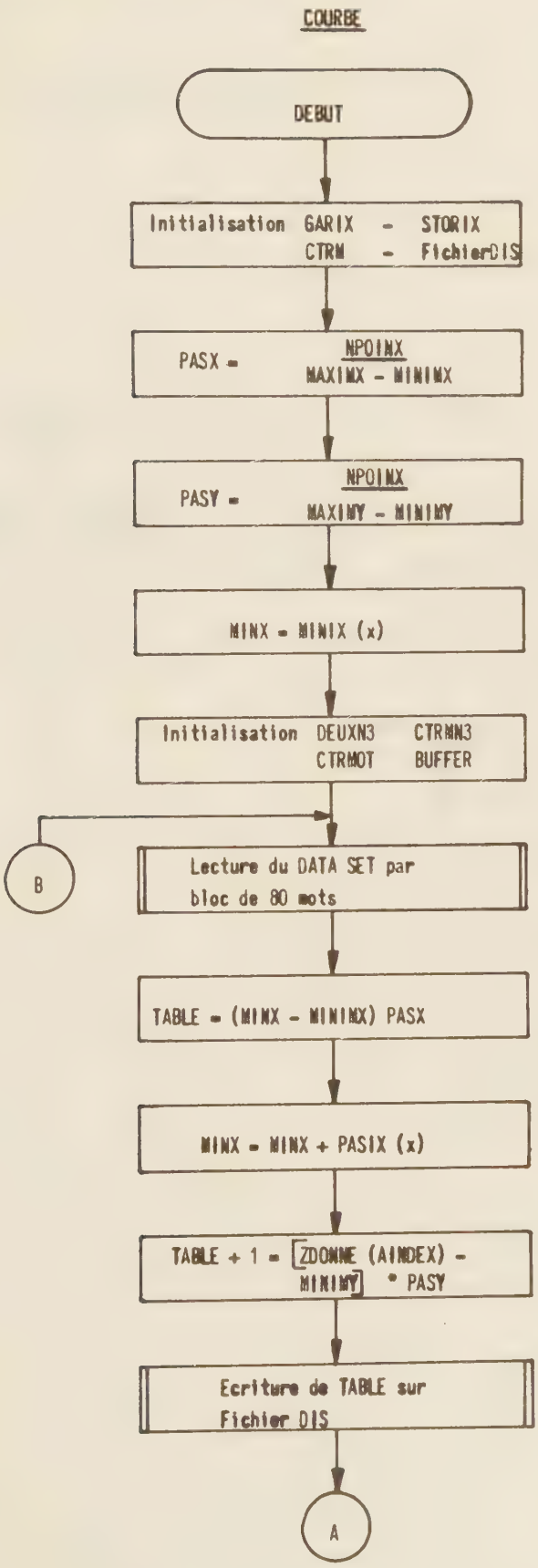


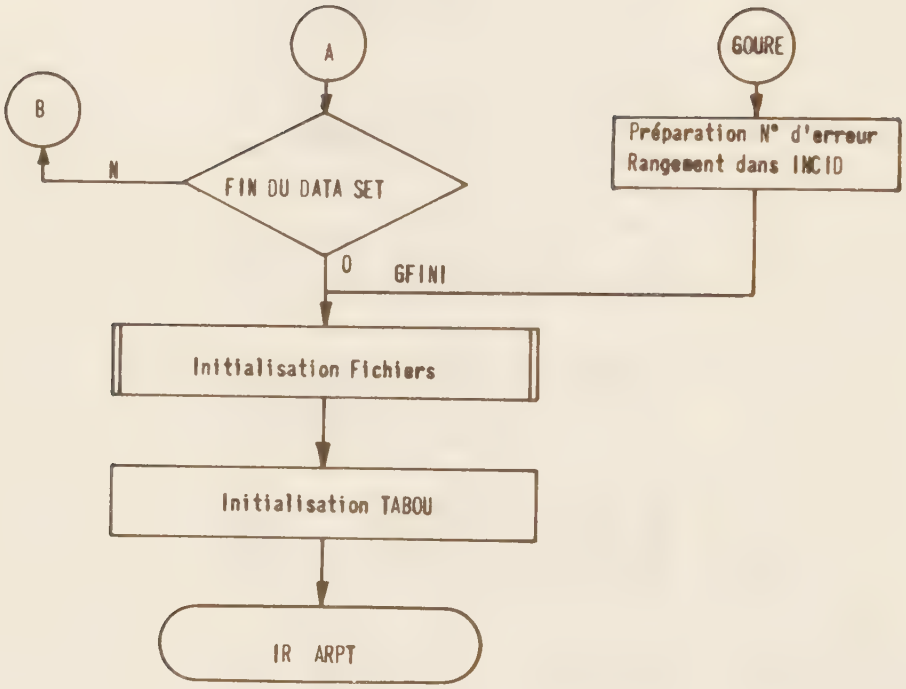
B - DESCRIPTION DETAILLEE

I - ORGANIGRAMME DE PRINCIPE

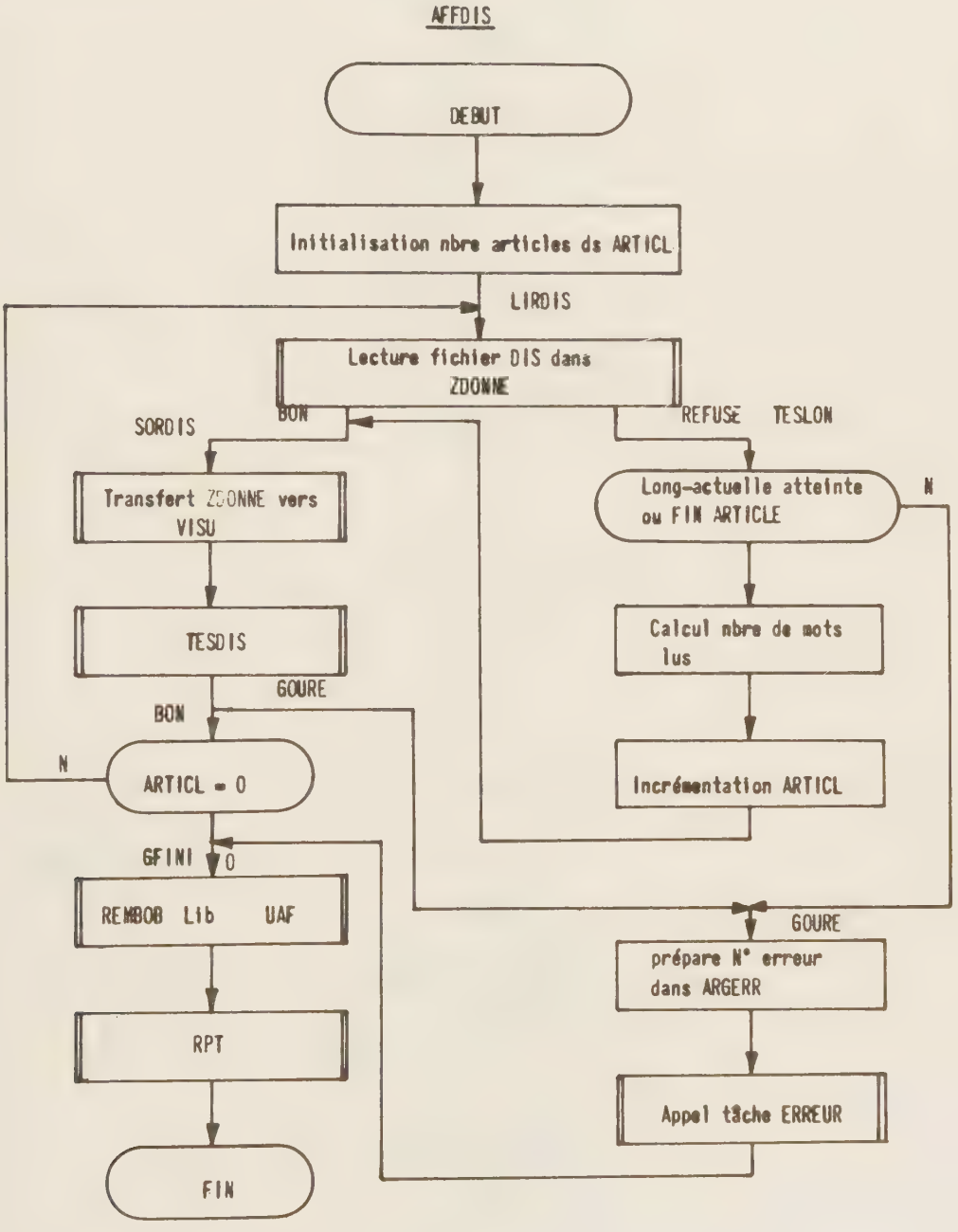


I - ORGANIGRAMME DE PRINCIPE



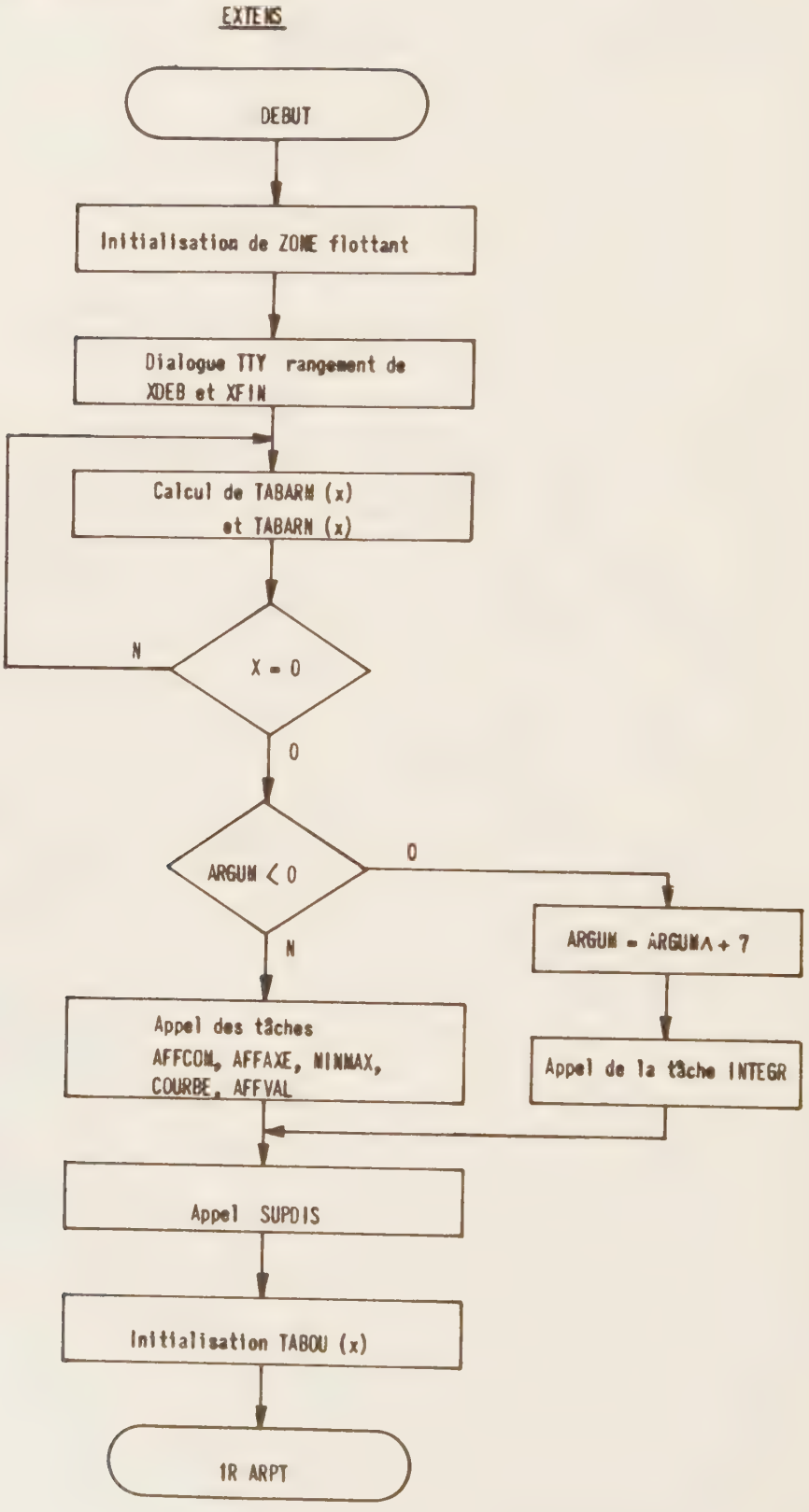

A
B
GOURE
N
FIN DU DATA SET
Préparation N° d'erreur
Rangement dans INCID
O
GFINI
Initialisation Fichiers
Initialisation TABOU
IR ARPT

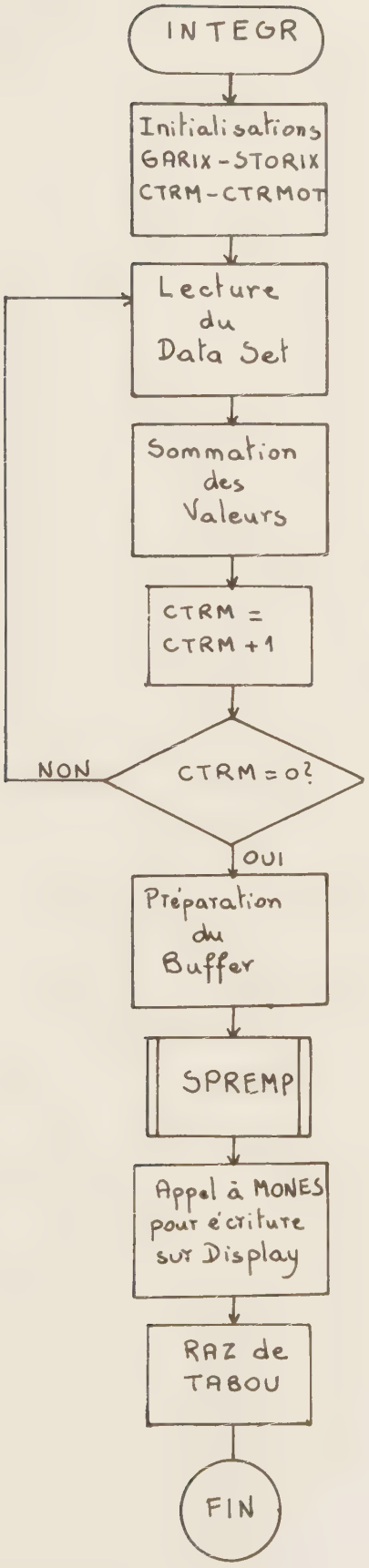
I - ORGANIGRAMME DE FONCTIONNEMENT



B - DESCRIPTION DETAILLEE

I - ORGANIGRAMME DE PRINCIPE




INTEGR
Initialisations GARIX-STORIX CTRM-CTRMOT
Lecture du Data Set
Sommation des Valeurs
CTRM = CTRM +1
NON
CTRM = 0?
OUI
Préparation du Buffer
SPREMP
Appel à MONES pour écriture sur Display
RAZ de TABOU
FIN

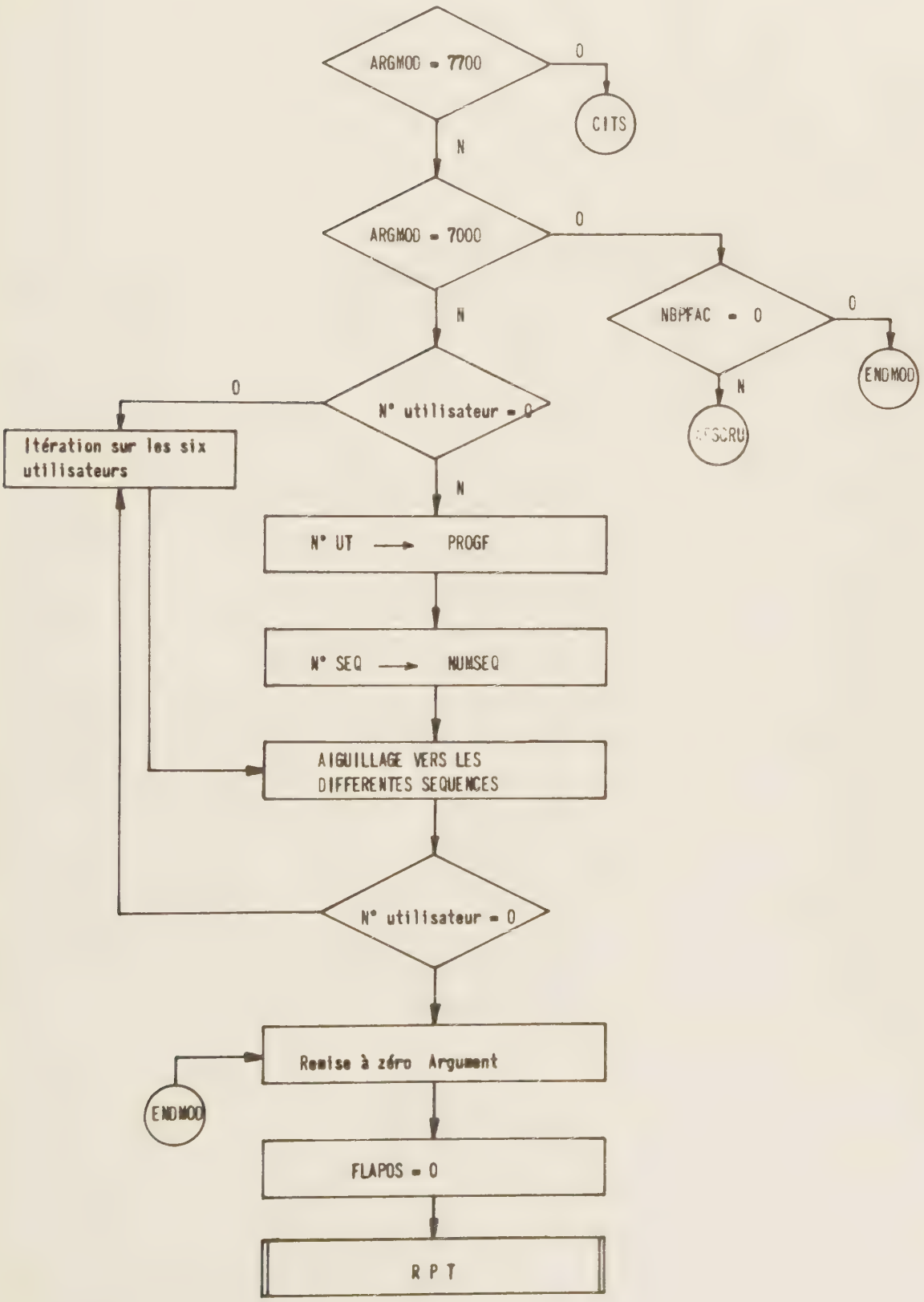
ANNEXE III

DOCUMENTS RELATIFS

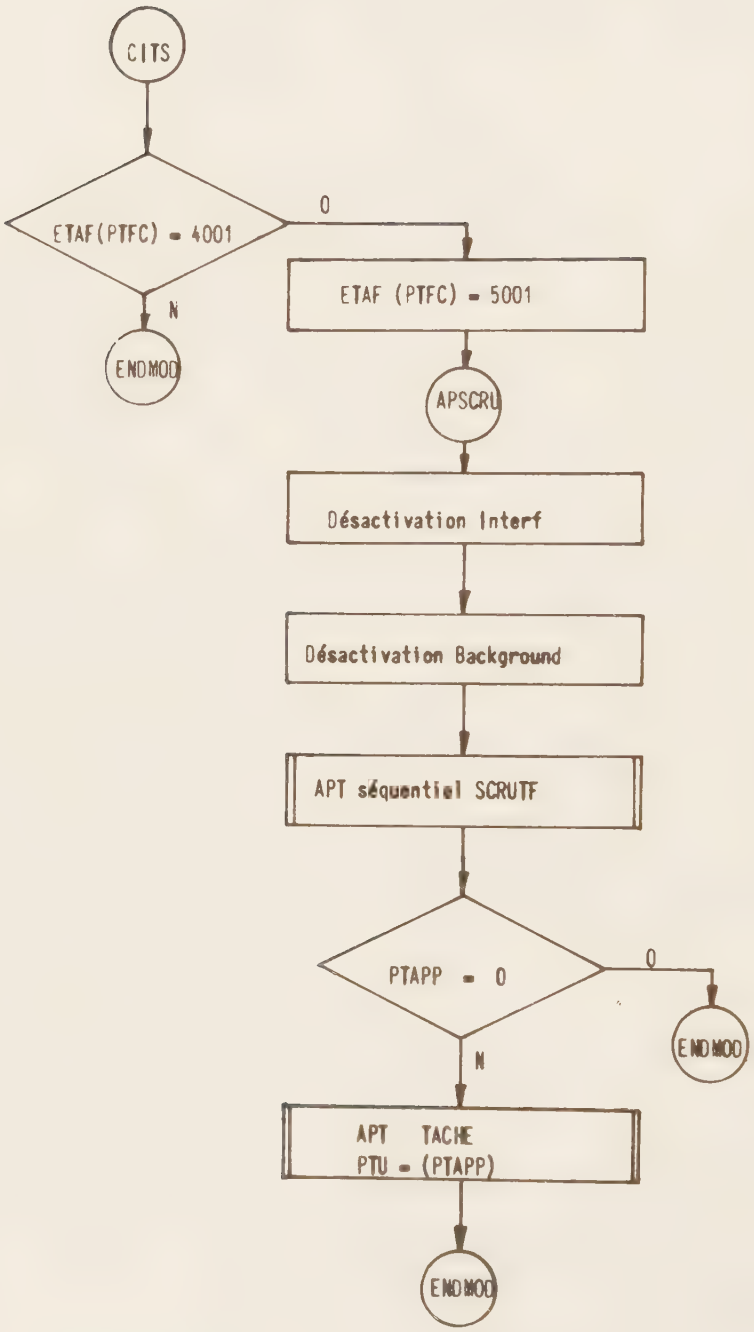
AU SYSTEME SYSFOR

III ORGANIGRAMMES DE MODULS

(A) ANALYSE DE L'ARGUMENT (ARGMOD)



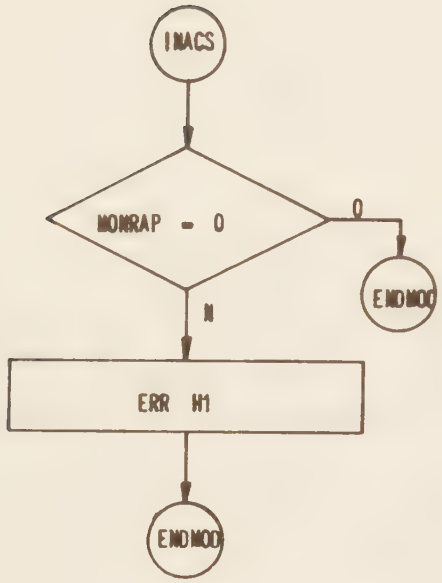
(B) ORGANIGRAMME CITS APSCRU

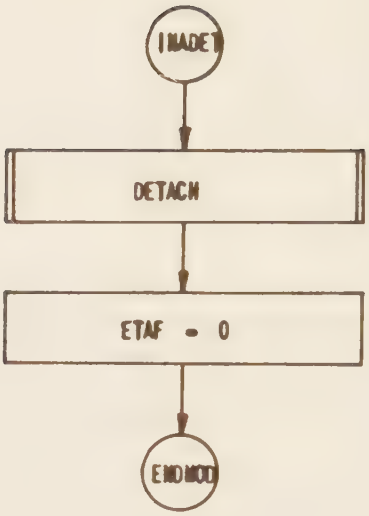


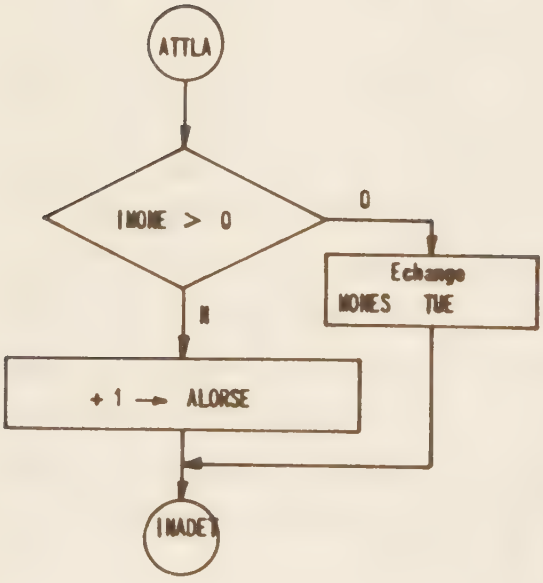
(C) AIGUILLAGE

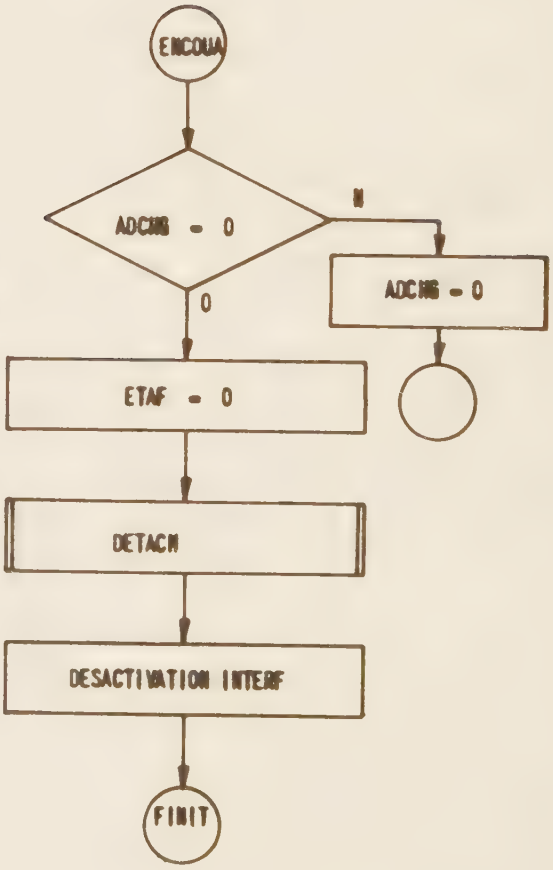
ETATS PROGRAMME / ACTIONS	NUL 0	INACTIF INTERROMPU 1	INACTIF ATTENTE LONGUE 2	EN COURS 4	ACTIF INTERROMPU 5	ACTIF ATTENTE LONGUE 6
0 ARRET	INACS	INADET	ATTLA	ENCOUA	INTERA	ATTLA
1 SUSPENSION	INACS	INACS	INACS	ENCOUS	INTERS	ATTLS
2 LANCEMENT	ENCOUL	INTERL	ATTLL	ERR H3	ERR H3	ERR H3
3 FIN ATTENTE LONGUE	ERR H0	ERR H0	FINATT	ERR H0	ERR H0	FINATT
4 FIN PROGRAMME	ERR H5	ERR H5	ERR H5	F P F	ERR H5	ERR H5
5 MISE ATTENTE LONGUE	ATTLA	ATTLS	ERR H4	BONUS	BONUS	BONUS
6 REACTIVATION	ENDMOD	ENDMOD	ENDMOD	REACTI	ENDMOD	ENDMOD

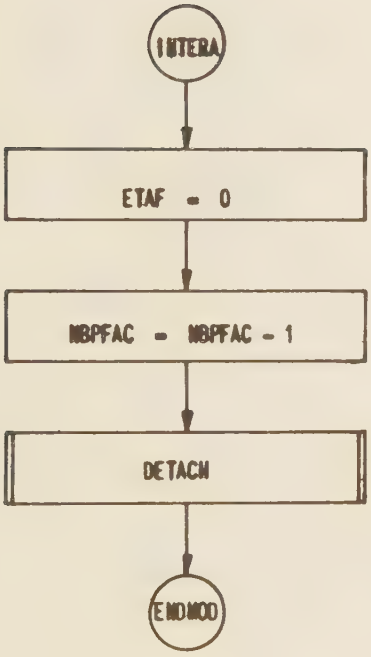
ARRET

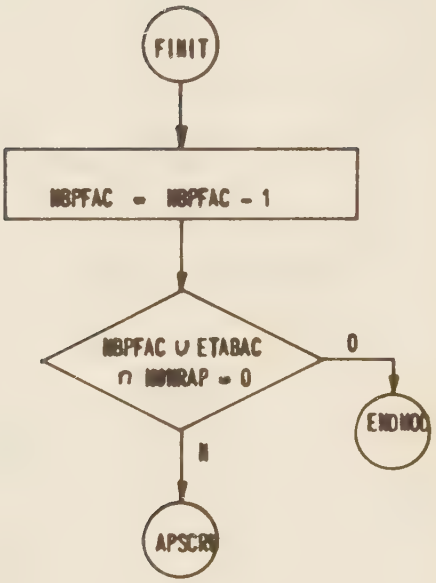

INACS
MONRAP = 0
0
N
ENDMOD
ERR M1
ENDMOD


INADET
DETACH
ETAF = 0
ENDMOD

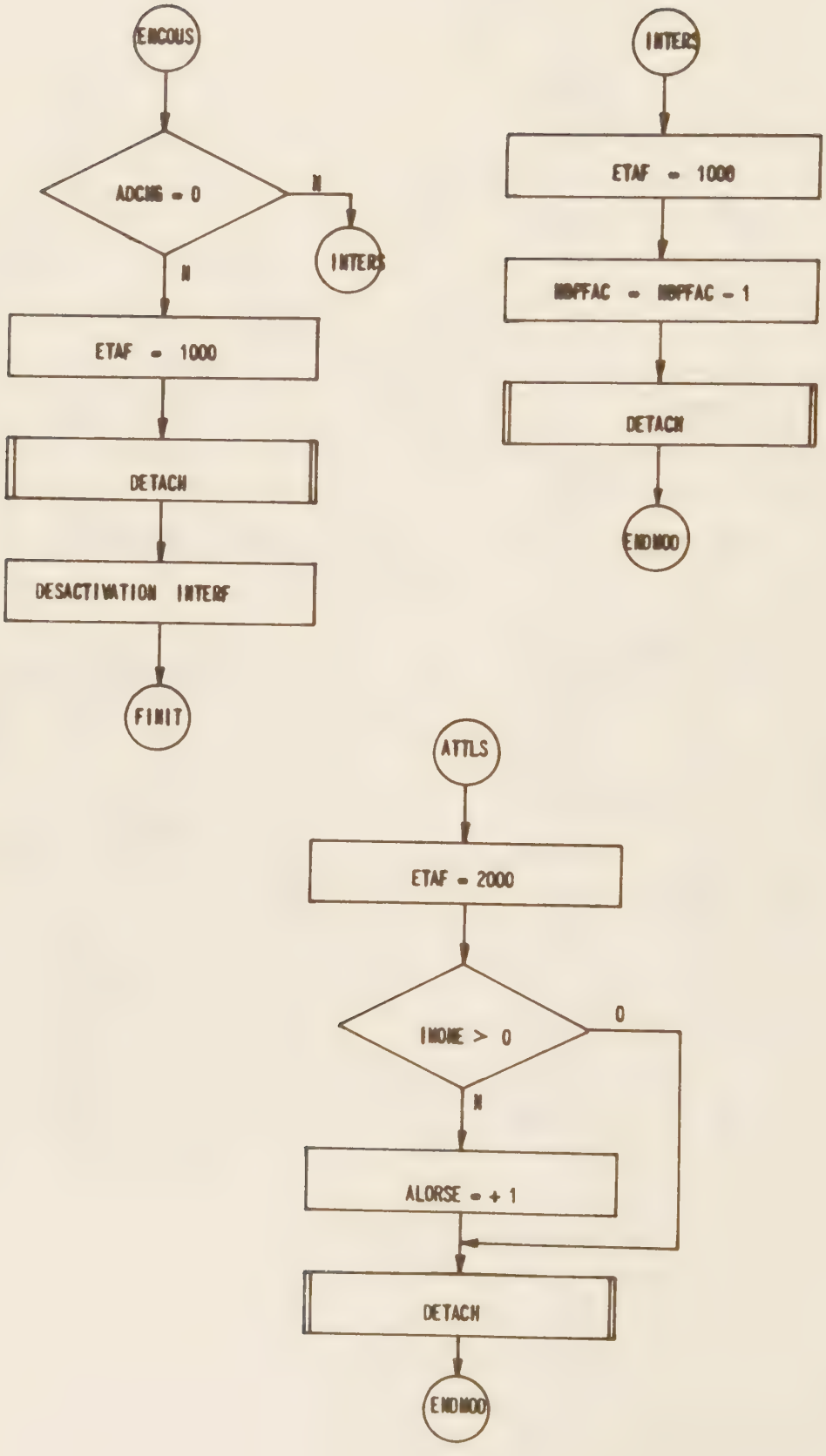

ATTLA
IMONE > 0
0
N
Echange
MONES TUE
+ 1 → ALORSE
INADET


ENCOUA
ADCMB = 0
N
0
ADCMB = 0
ETAF = 0
DETACH
DESACTIVATION INTERF
FINIT

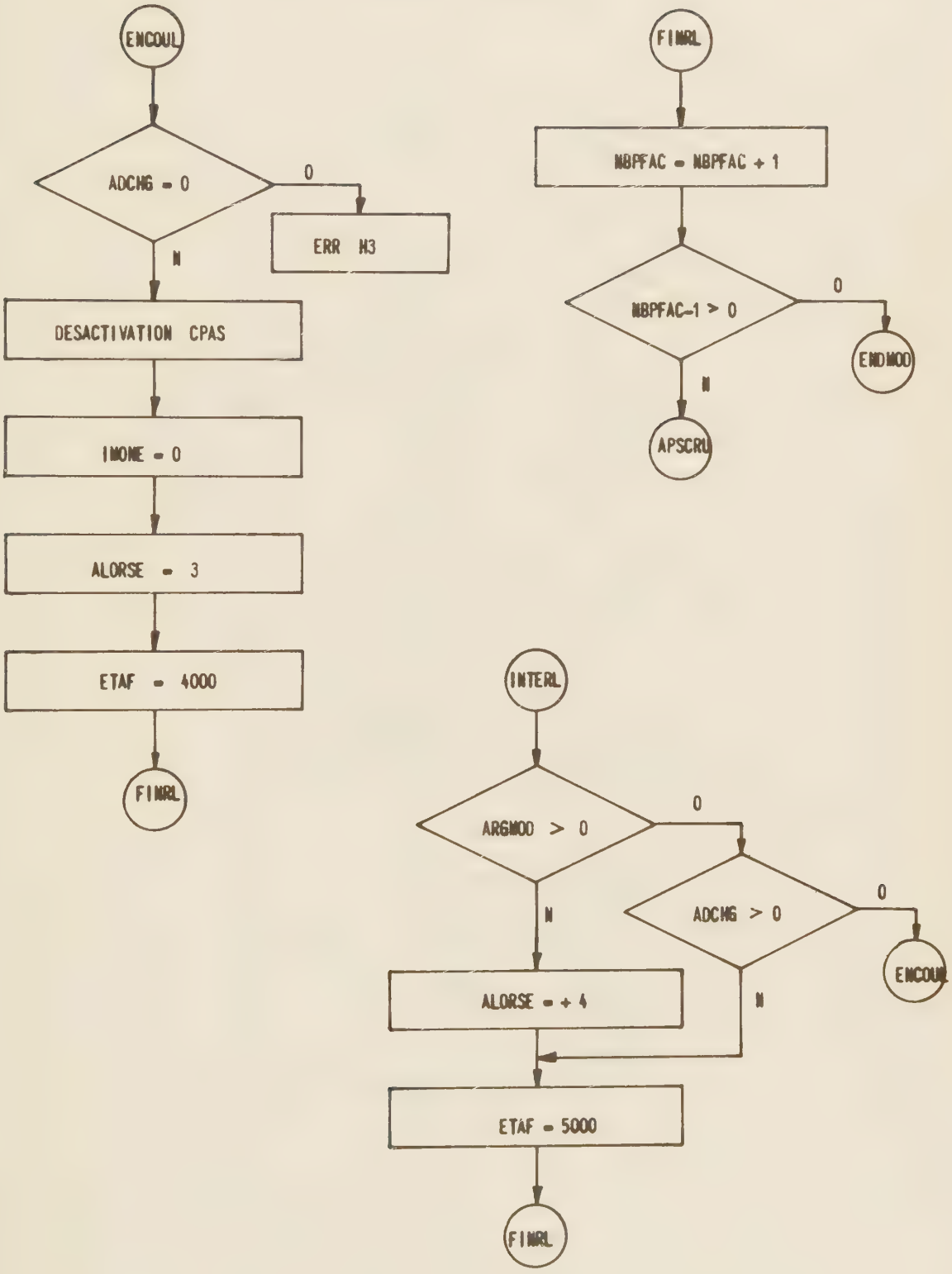

INTERA
ETAF = 0
NBPFAC = NBPFAC - 1
DETACH
ENDMOD

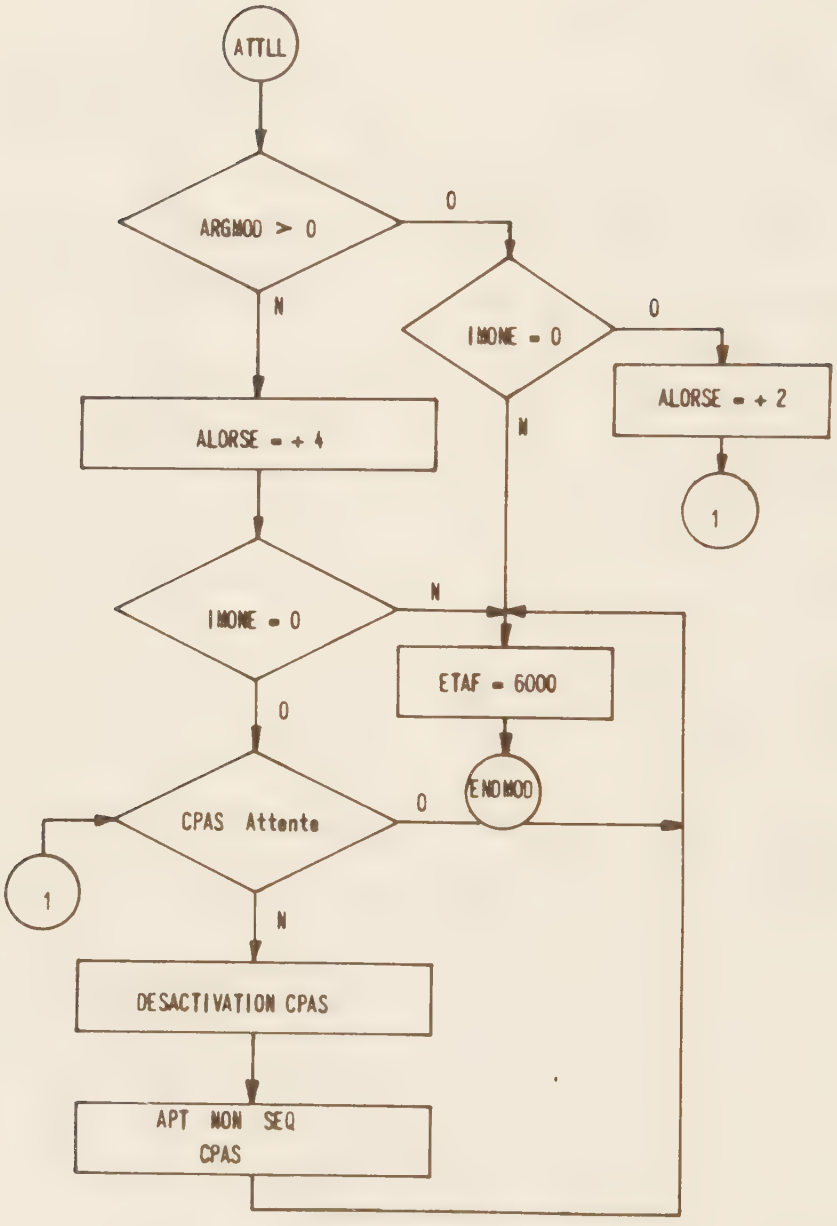

FINIT
NBPFAC = NBPFAC - 1
NBPFAC ∪ ETABAC
∩ NBMRAP = 0
0
N
ENDMOD
APSCR

SUSPENSION

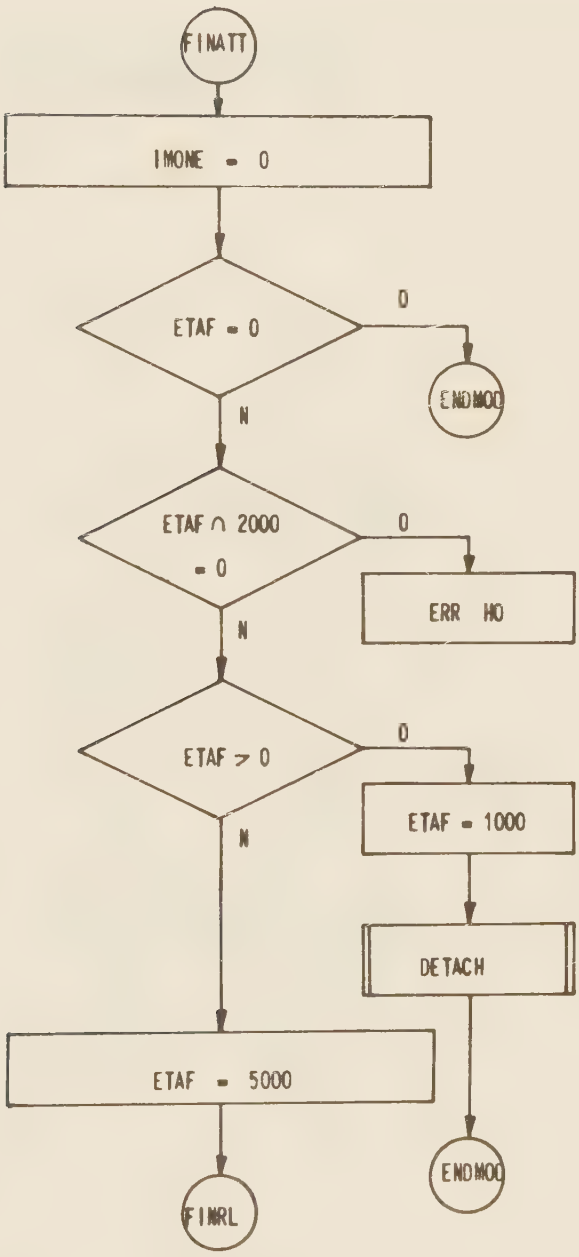

ENCOUS
ADCMS = 0
N
INTERS
N
ETAF = 1000
DETACH
DESACTIVATION INTERF
FINIT
INTERS
ETAF = 1000
NBPFAC = NBPFAC - 1
DETACH
ENDMOD
ATTLS
ETAF = 2000
INOME > 0
O
N
ALORSE = + 1
DETACH
ENDMOD

LANCEMENT

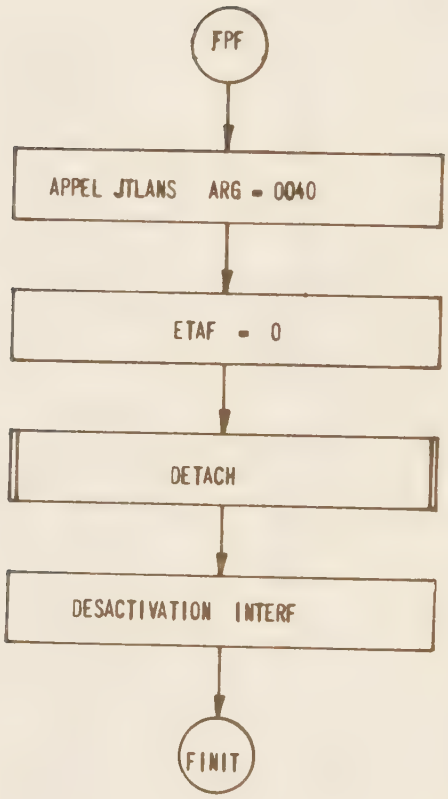

ENCOUL
ADCHG = 0
0
ERR M3
N
DESACTIVATION CPAS
IMONE = 0
ALORSE = 3
ETAF = 4000
FINRL
FINRL
NBPFAC = NBPFAC + 1
NBPFAC-1 > 0
0
ENDMOD
N
APSCRU
INTERL
ARGMOD > 0
0
ADCHG > 0
0
ENCOUL
N
ALORSE = + 4
N
ETAF = 5000
FINRL


ATTLL
ARGMOD > 0
O
N
IMONE = 0
O
ALORSE = + 2
1
N
ALORSE = + 4
IMONE = 0
N
O
ETAF = 6000
ENDMOD
CPAS Attente
O
1
N
DESACTIVATION CPAS
APT NON SEQ
CPAS

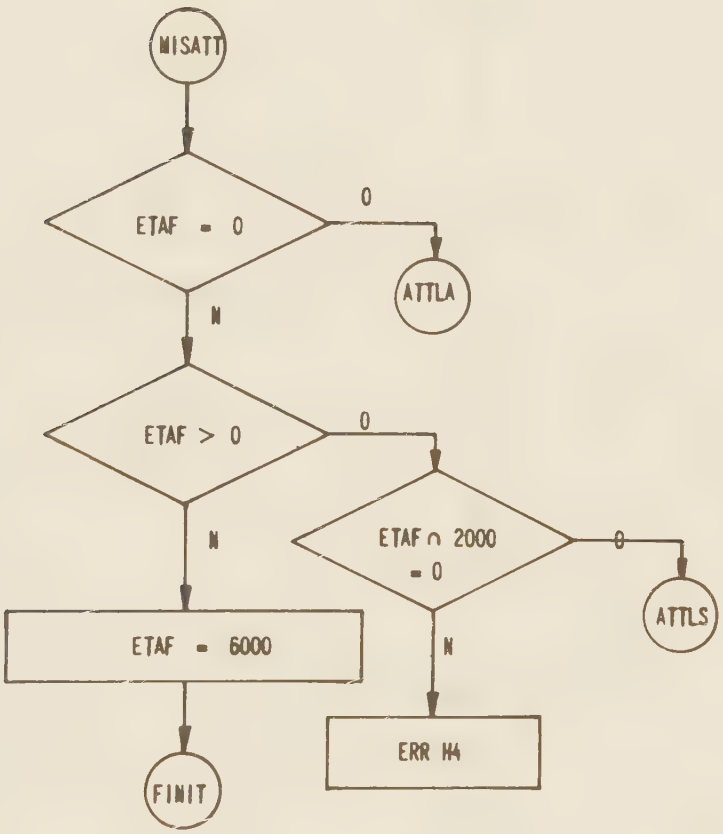
FIN ATTENTE LONGUE



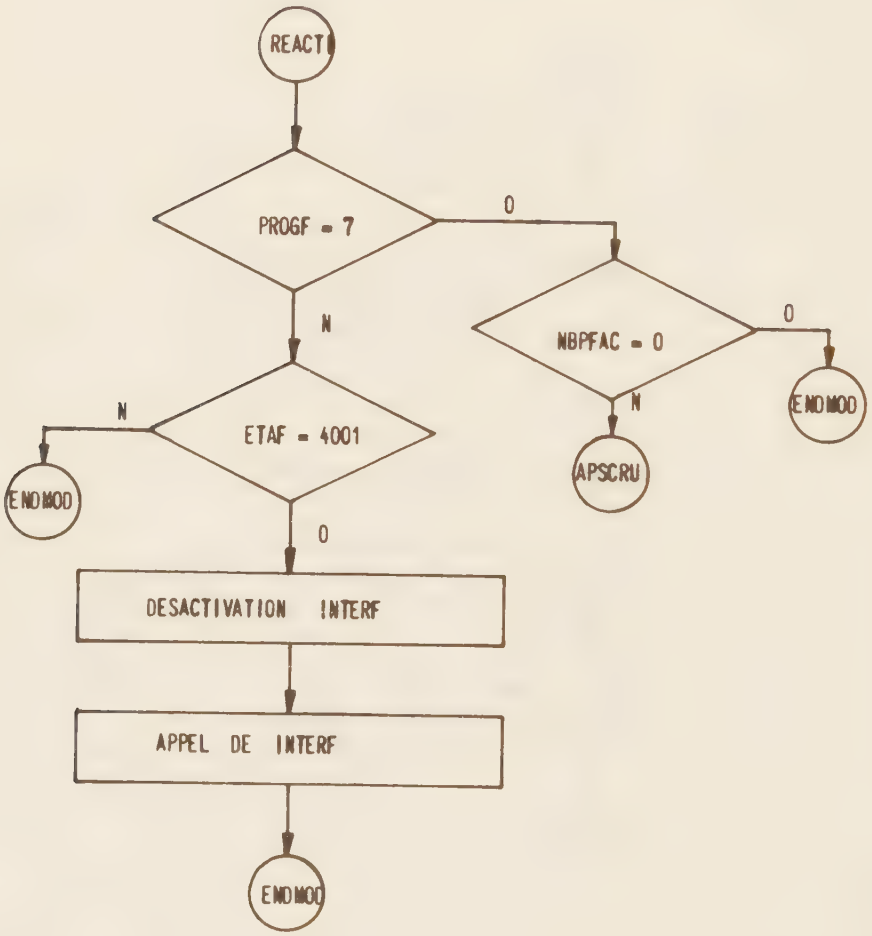
FIN PROGRAMME FORTRAN



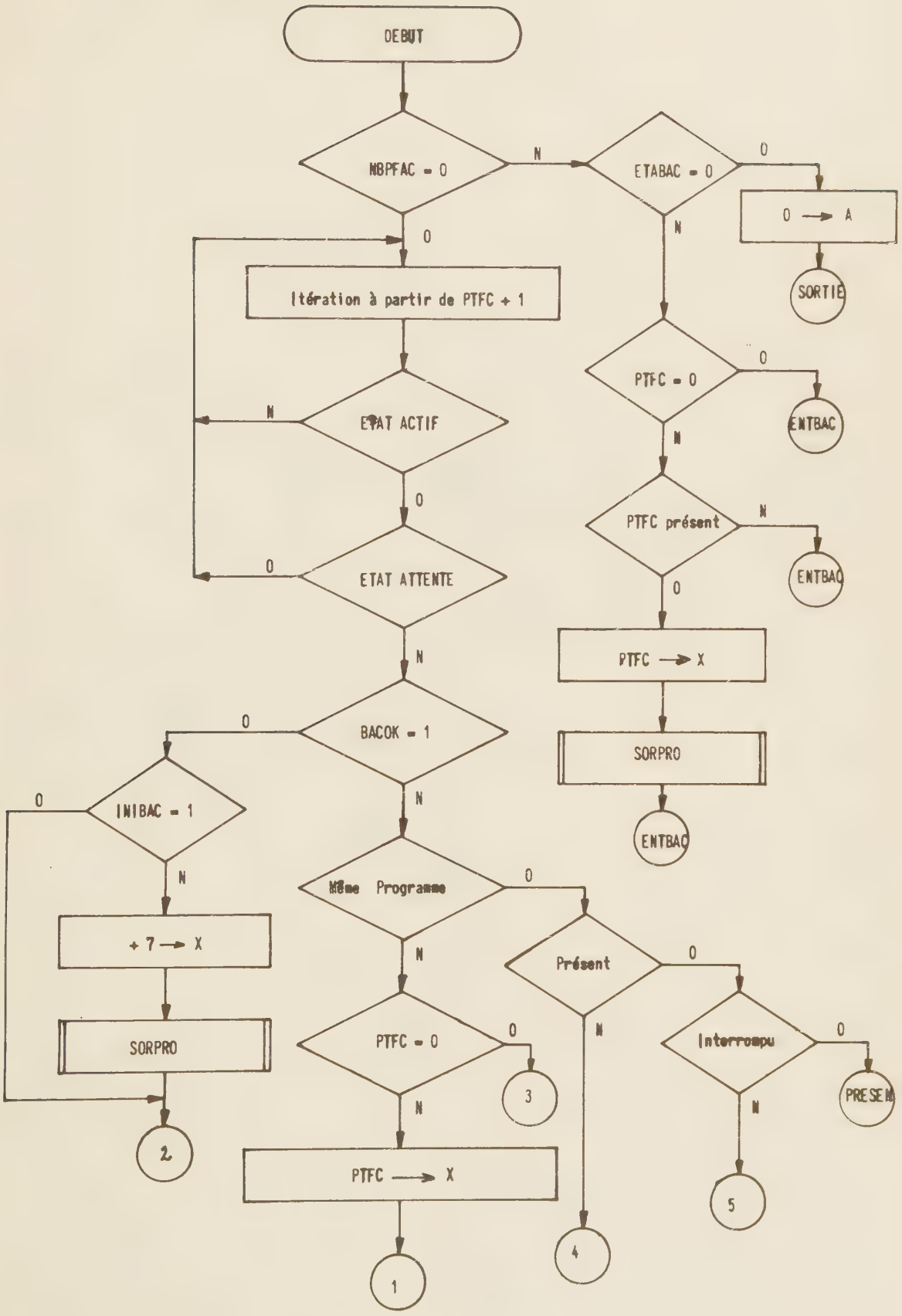
MISE EN ATTENTE LONGUE

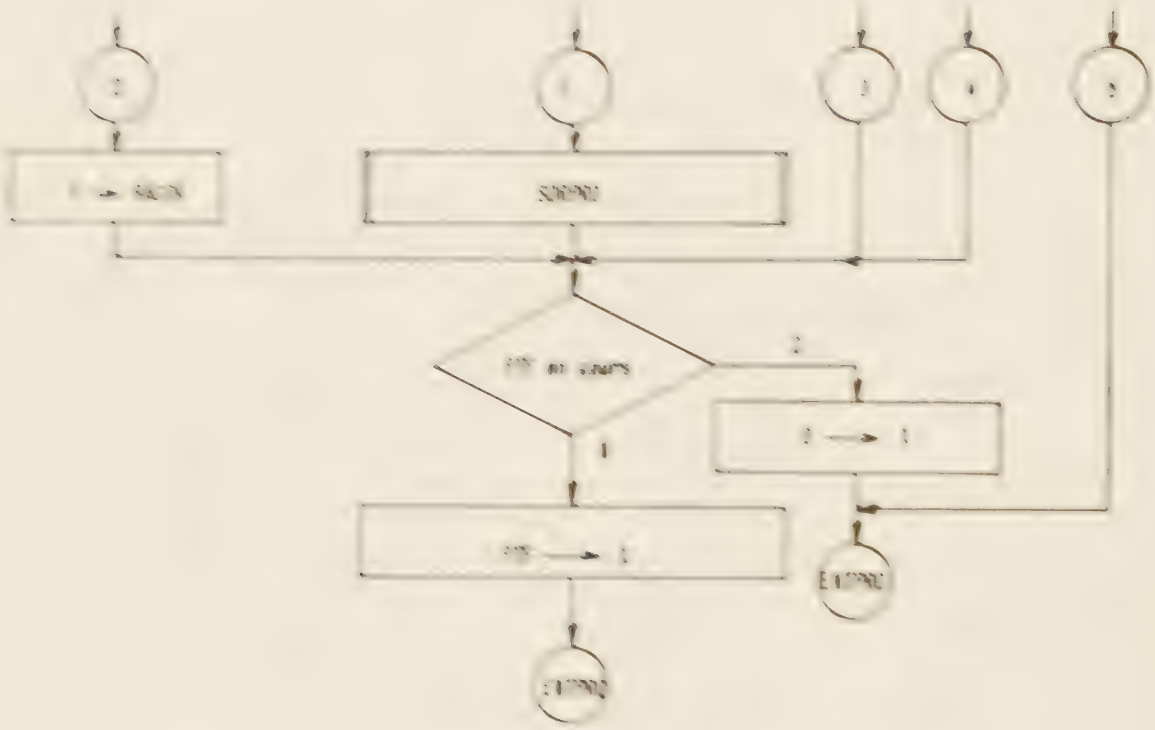


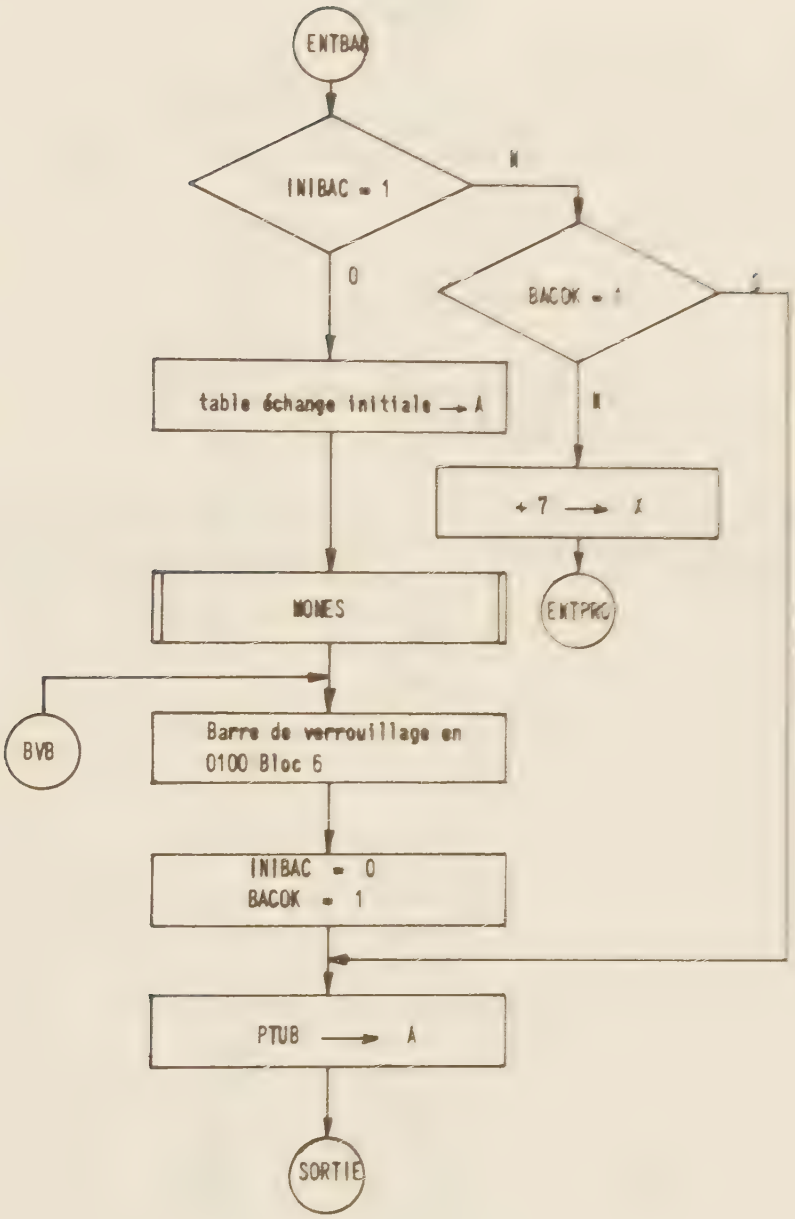
REACTIVATION APRES ECHANGE DISQUE

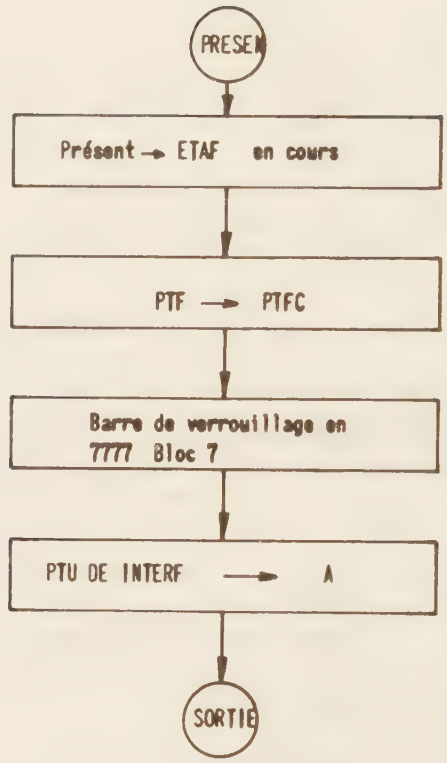


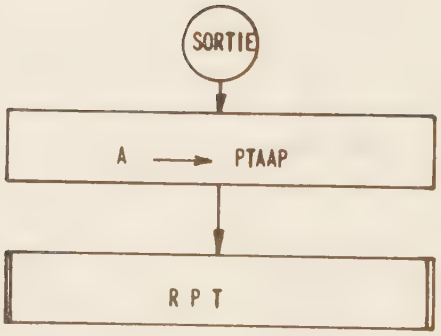
III - ORGANIGRAMMES

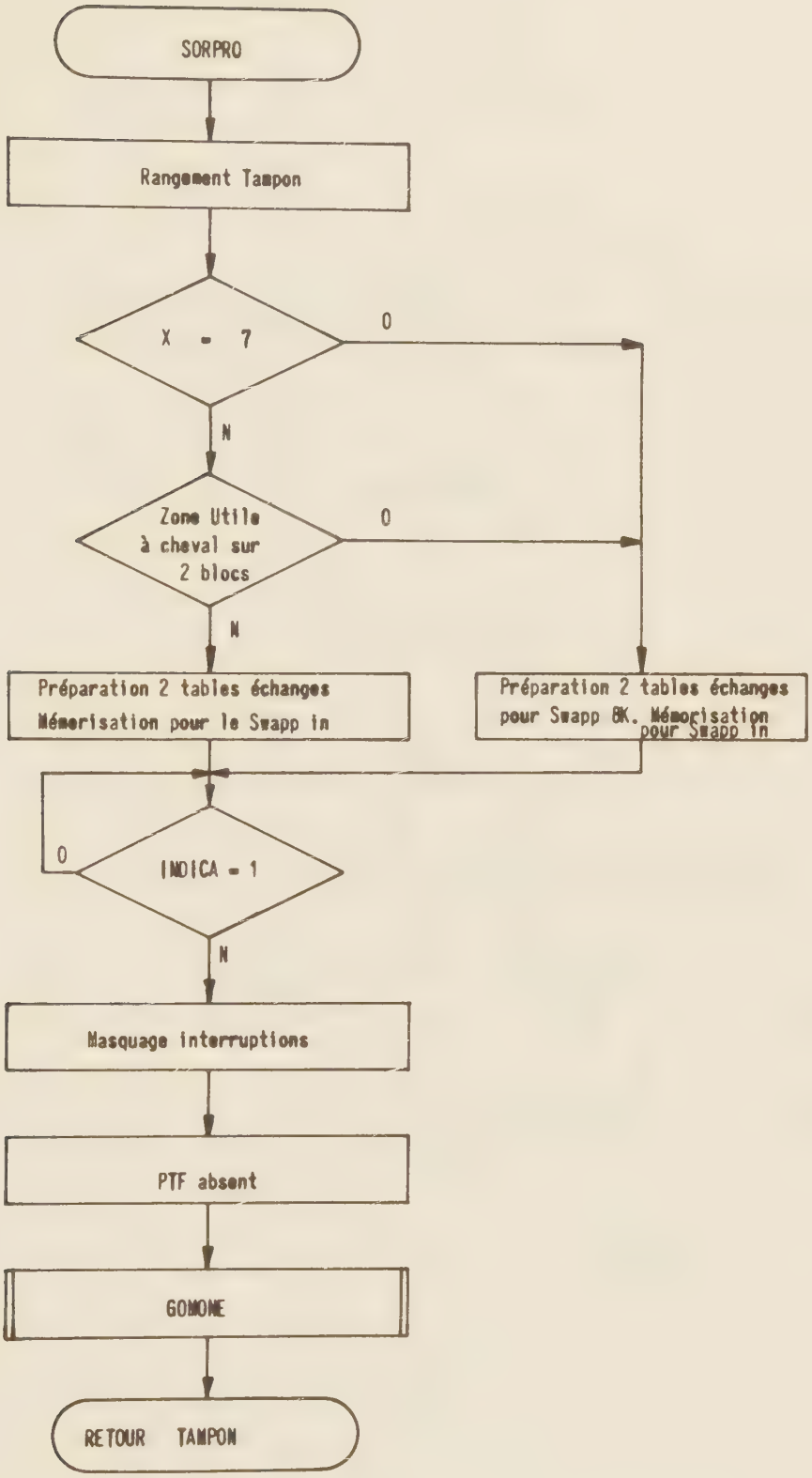


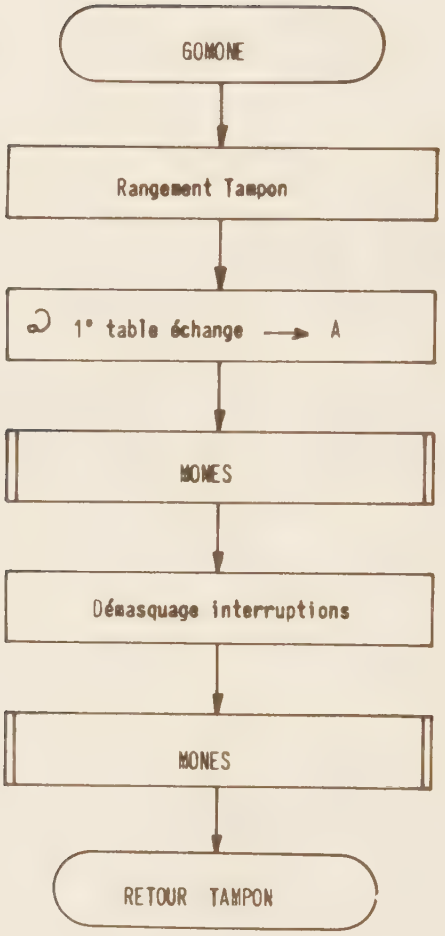


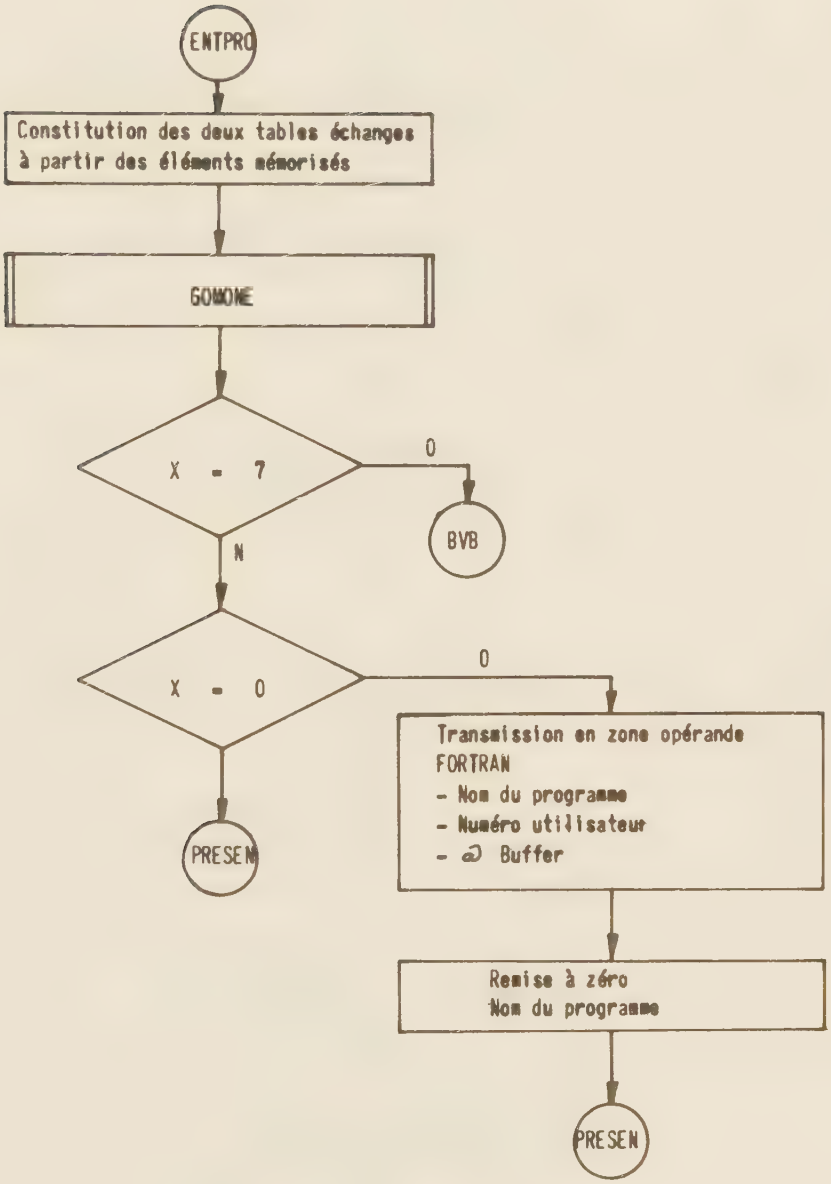

ENTBAC
INIBAC = 1
N
O
BACOK = 1
O
table échange initiale → A
N
+ 7 → X
MONES
ENTPRC
BVB
Barre de verrouillage en
0100 Bloc 6
INIBAC = 0
BACOK = 1
PTUB → A
SORTIE


PRESEN
Présent → ETAF en cours
PTF → PTFC
Barre de verrouillage en
7777 Bloc 7
PTU DE INTERF → A
SORTIE

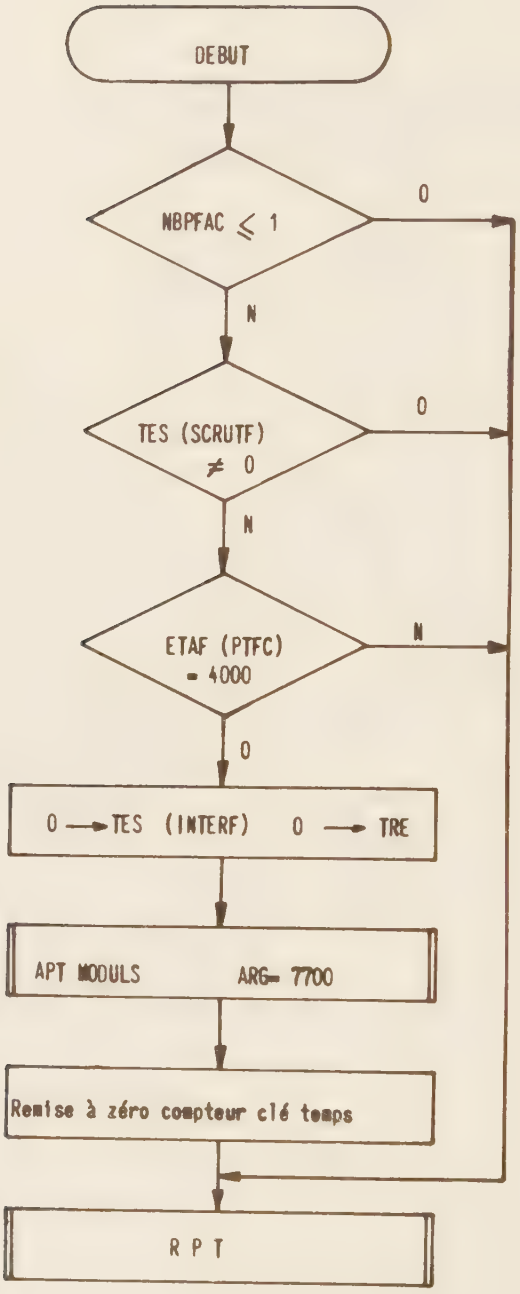

SORTIE
A → PTAAP
R P T


SORPRO
Rangement Tampon
X = 7
O
N
Zone Utile
à cheval sur
2 blocs
O
N
Préparation 2 tables échanges
Mémorisation pour le Swapp in
Préparation 2 tables échanges
pour Swapp BK. Mémorisation
pour Swapp in
O
INDICA = 1
N
Masquage interruptions
PTF absent
GOMONE
RETOUR TAMPON

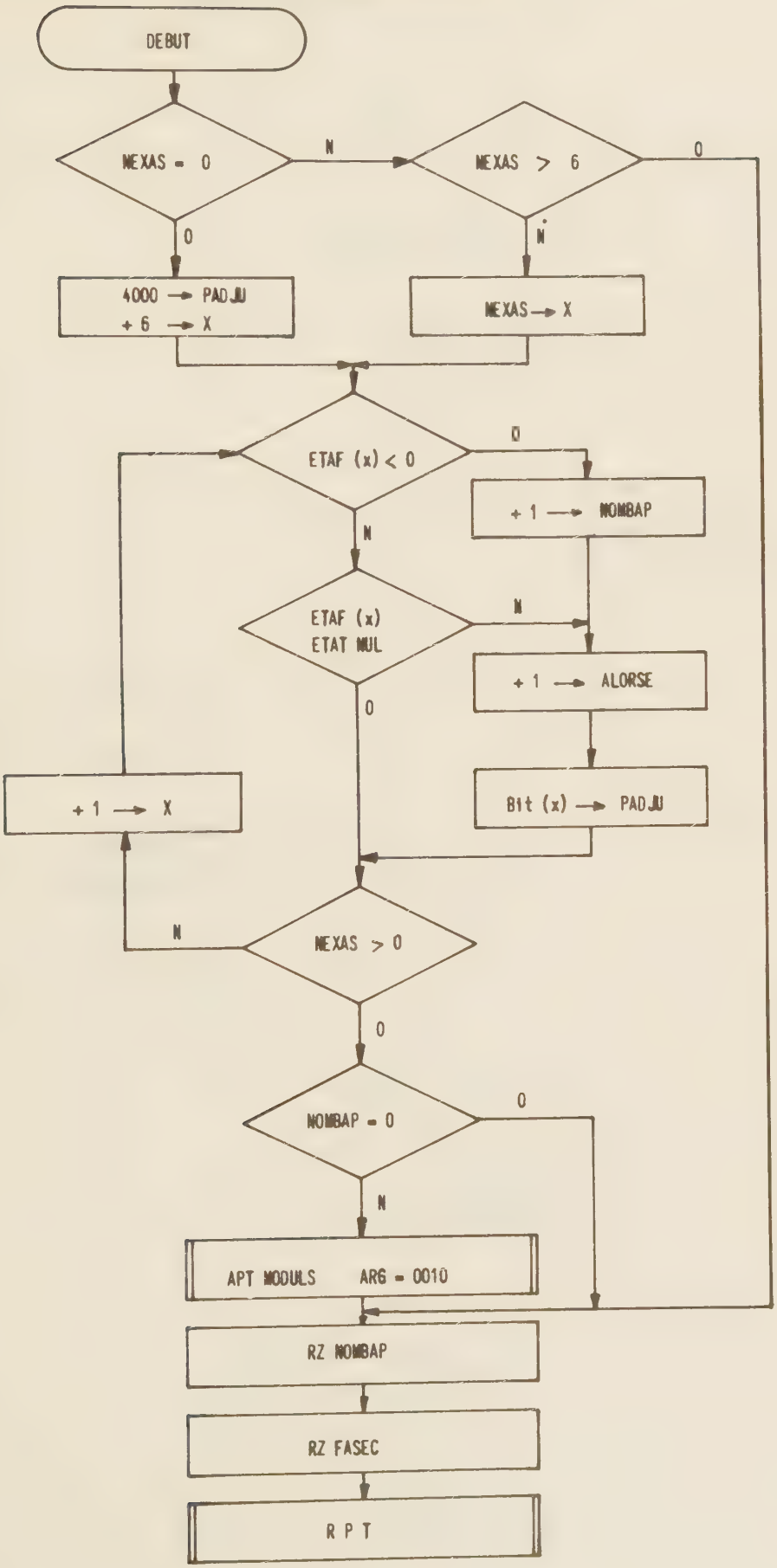

GOMONE
Rangement Tampon
1° table échange → A
MONES
Démasquage interruptions
MONES
RETOUR TAMPON


ENTPRO
Constitution des deux tables échanges
à partir des éléments mémorisés
GOMONE
X = 7
0
BVB
N
X = 0
0
PRESEN
Transmission en zone opérande
FORTRAN
- Nom du programme
- Numéro utilisateur
- @ Buffer
Remise à zéro
Nom du programme
PRESEN

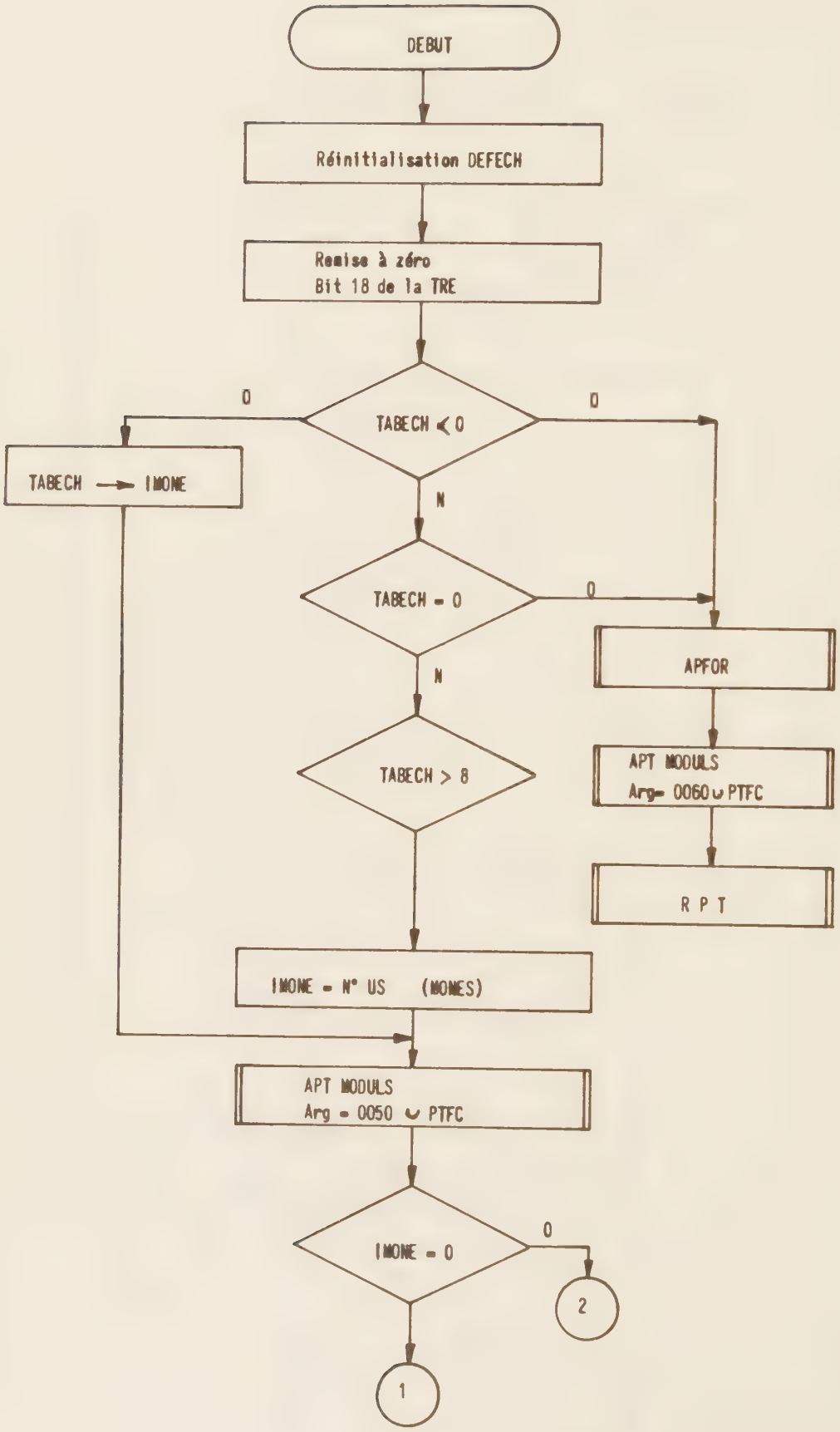
III - ORGANIGRAMME

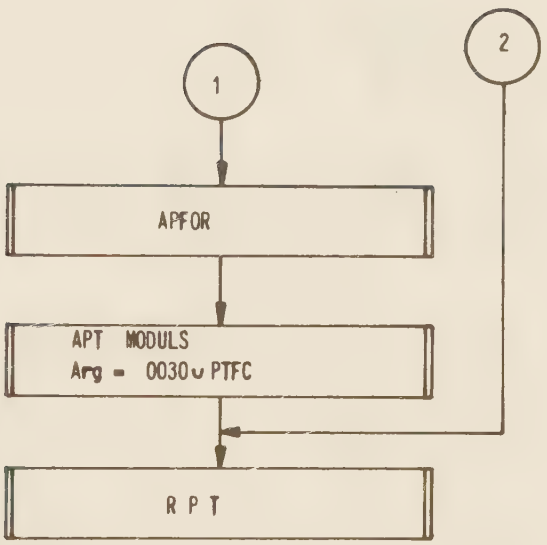


III - ORGANIGRAMME

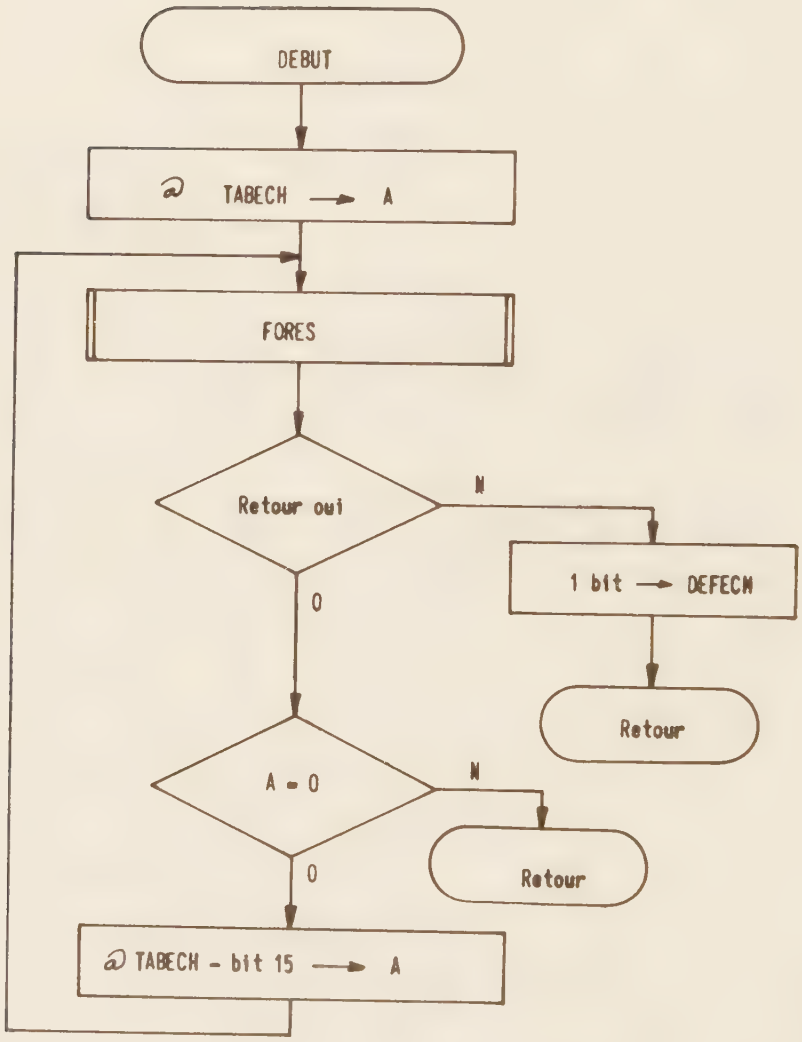


III - ORGANIGRAMME

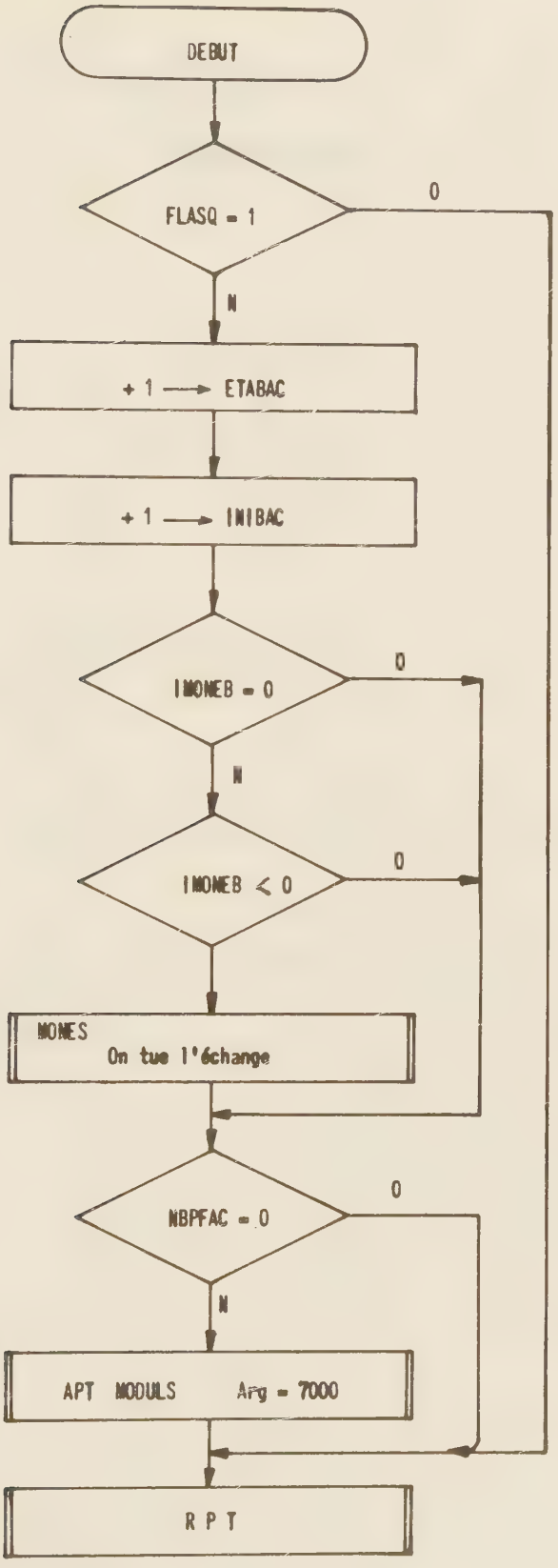



1
2
APFOR
APT MODULS
Arg = 0030 ∪ PTFC
R P T

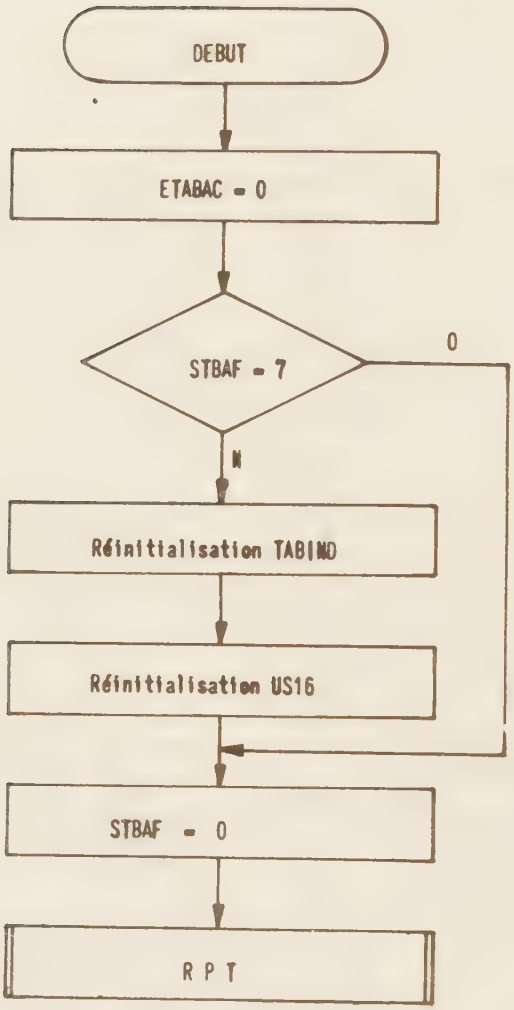
SOUS PROGRAMME APFOR



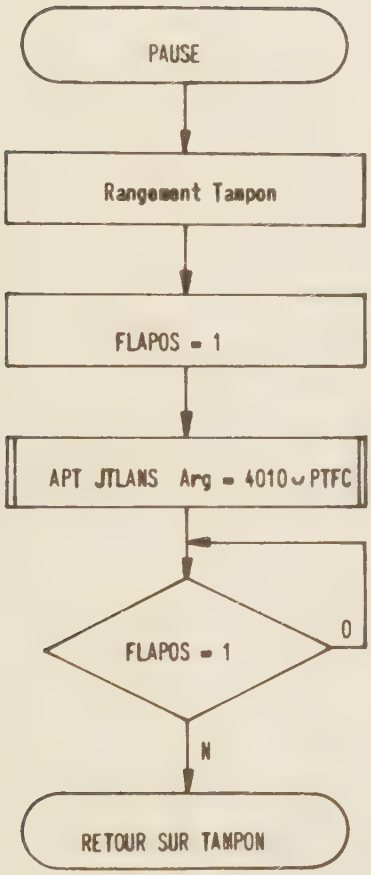
III - ORGANIGRAMME

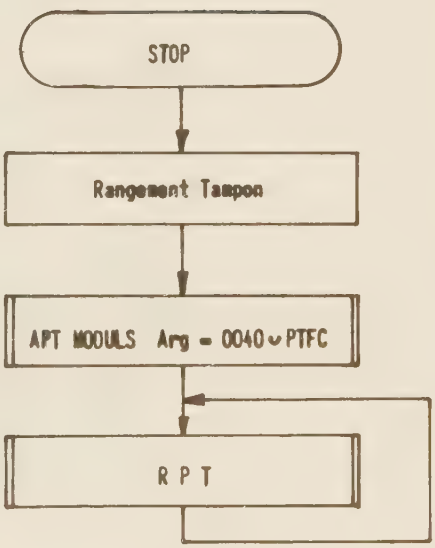


III - ORGANIGRAMME

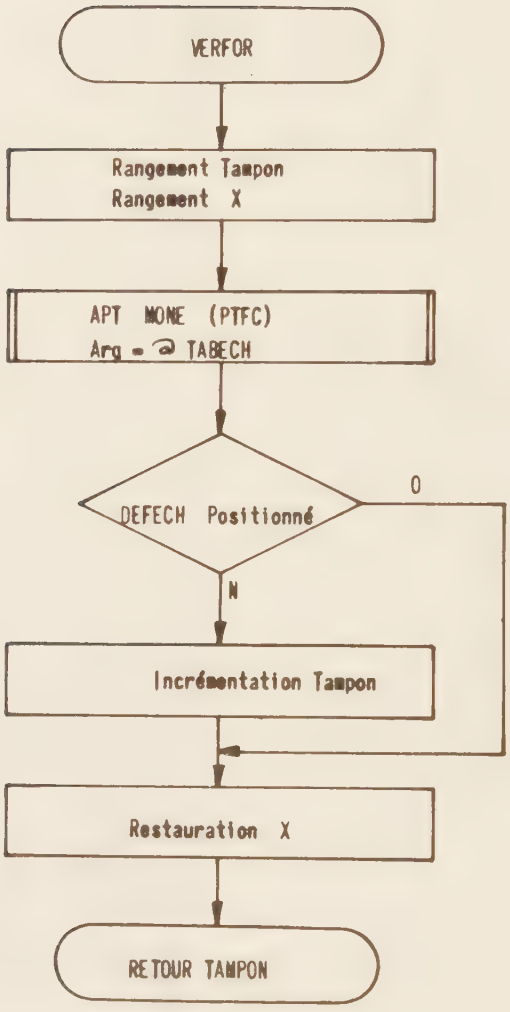


III - ORGANIGRAMMES

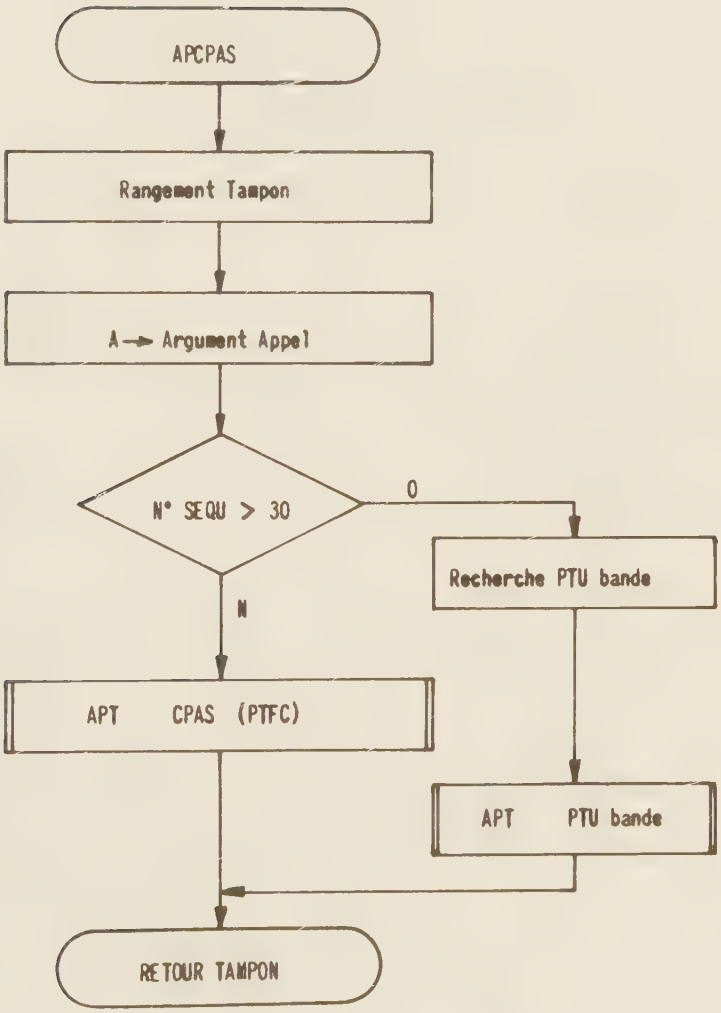




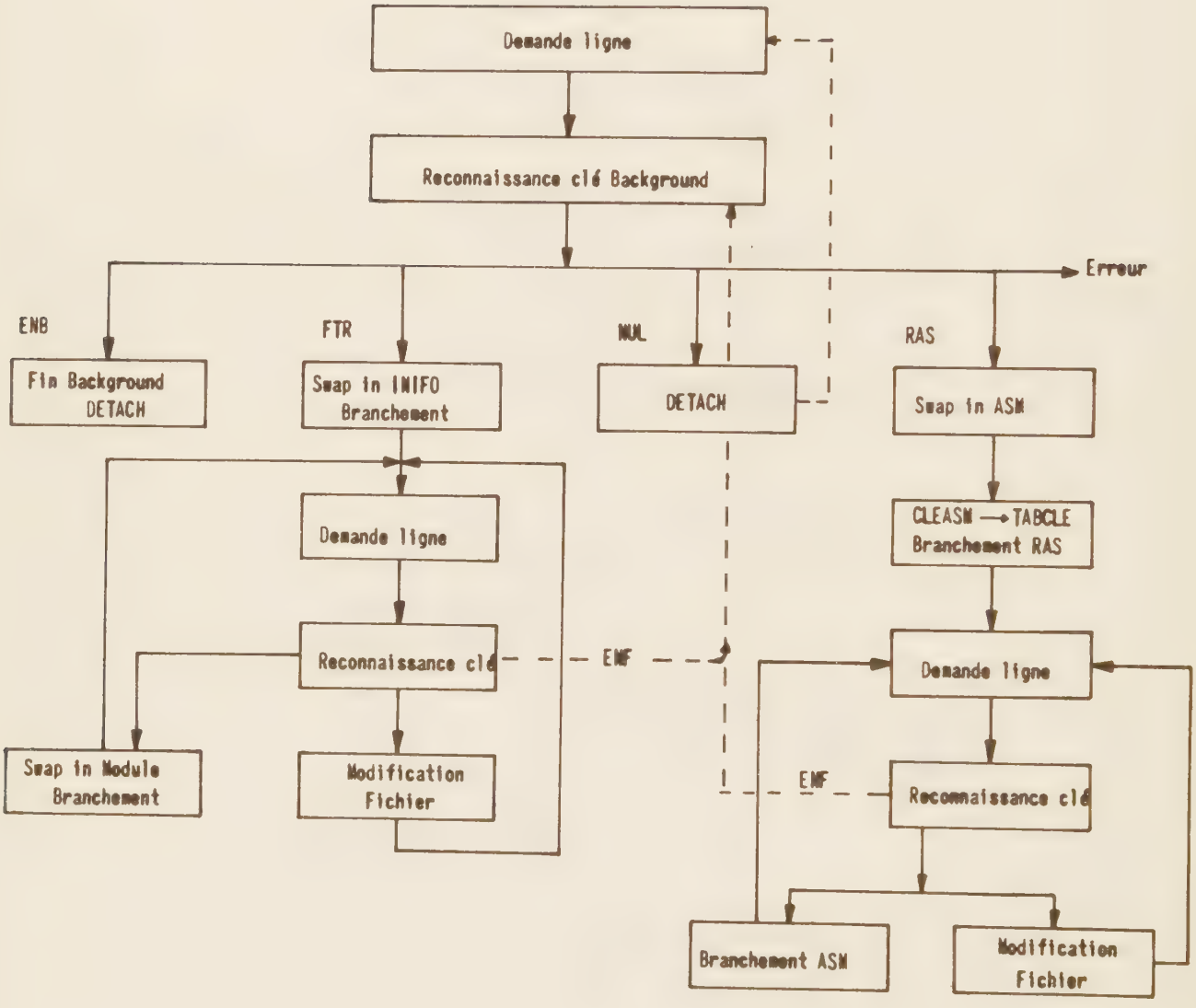
TRAITEMENT E/S



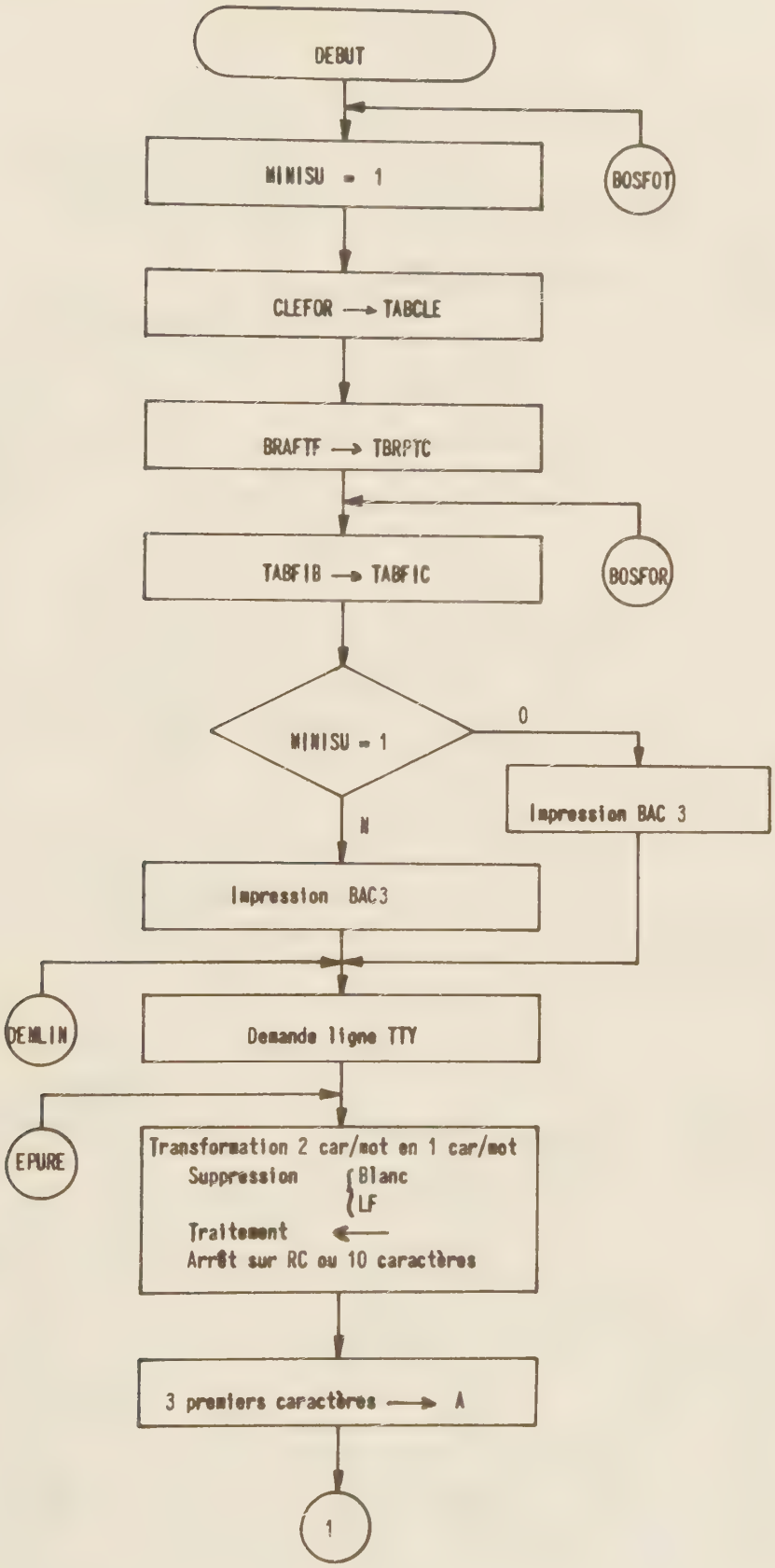
DEMANDE CAMAC

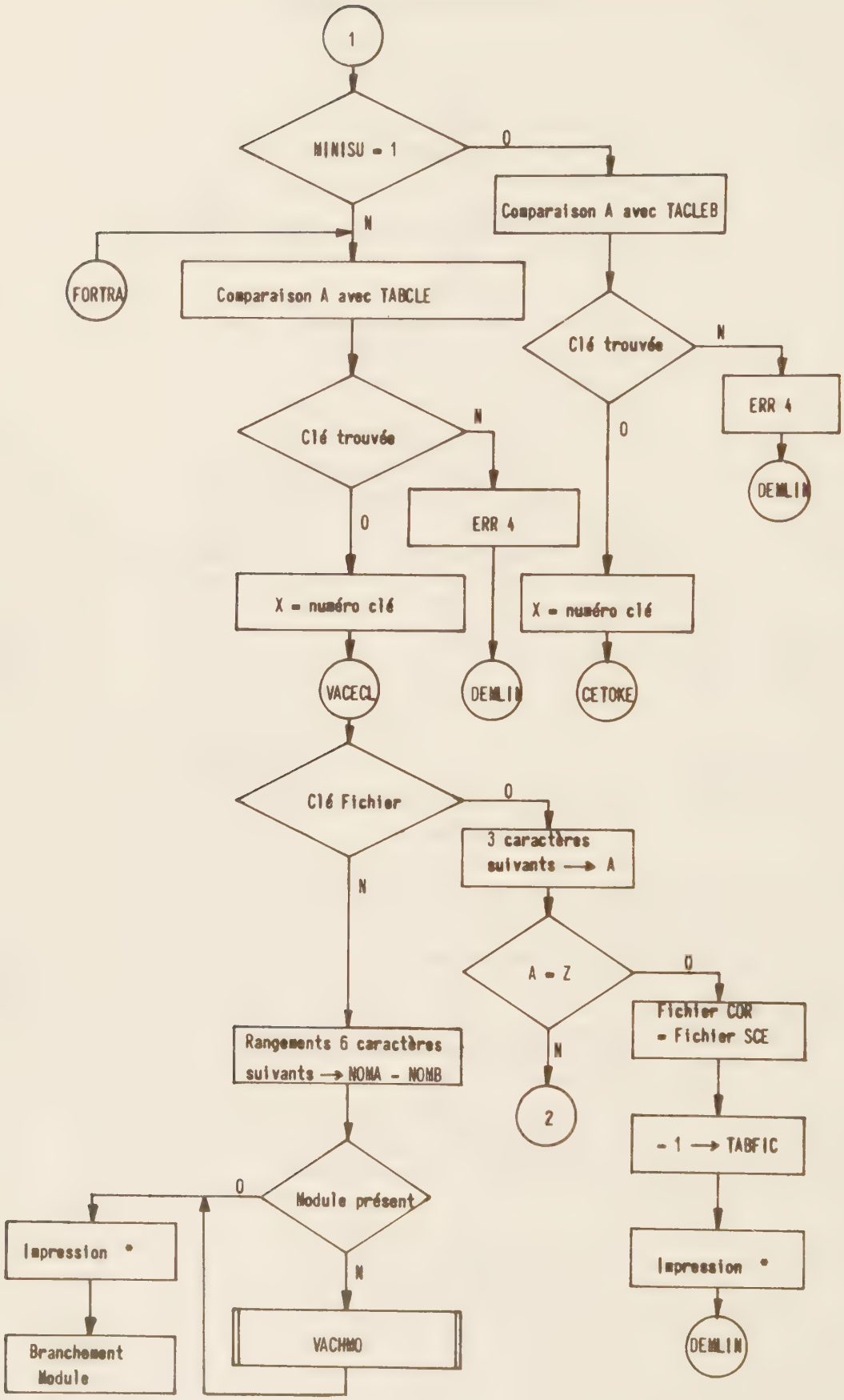


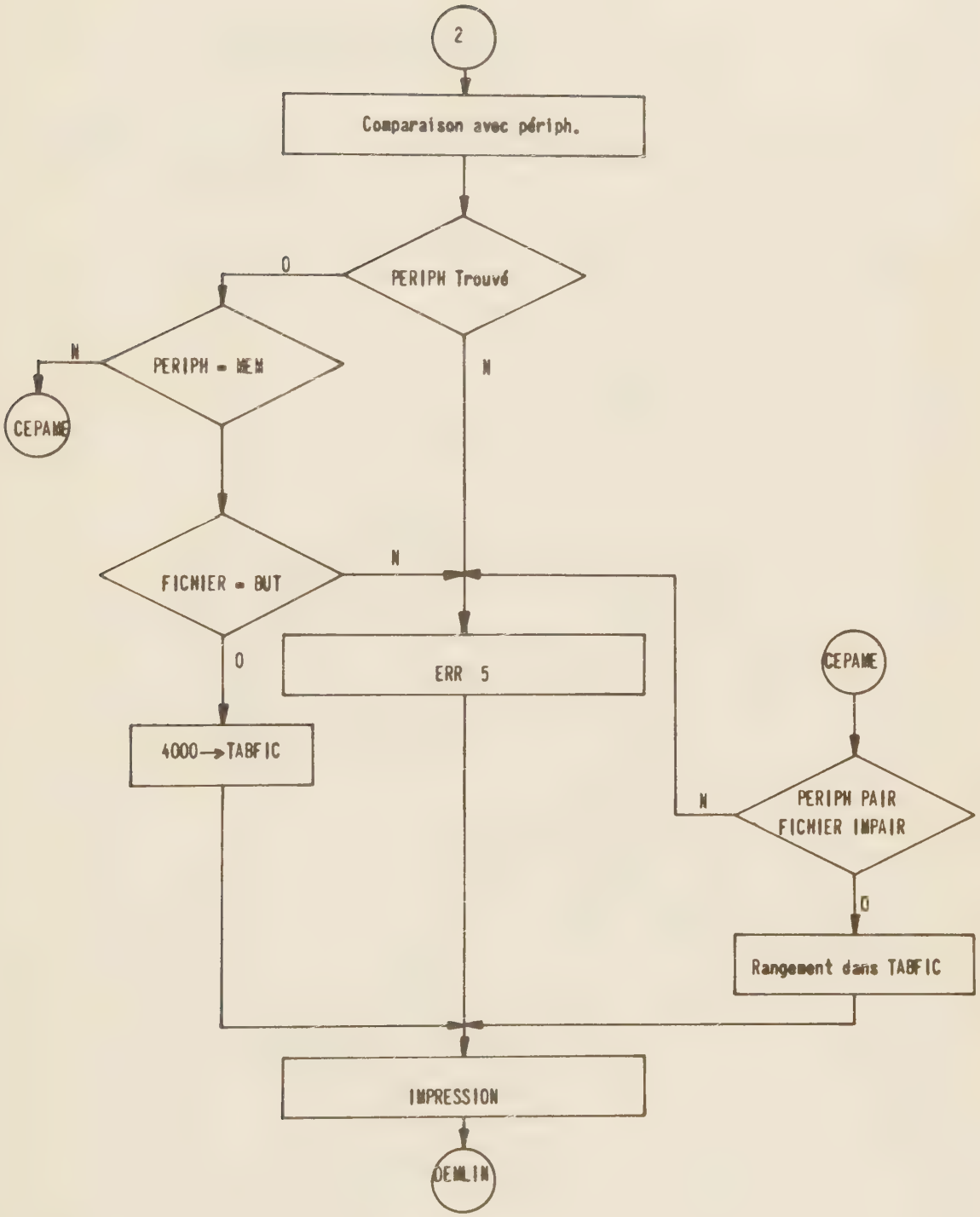
III - ORGANIGRAMME GENERAL

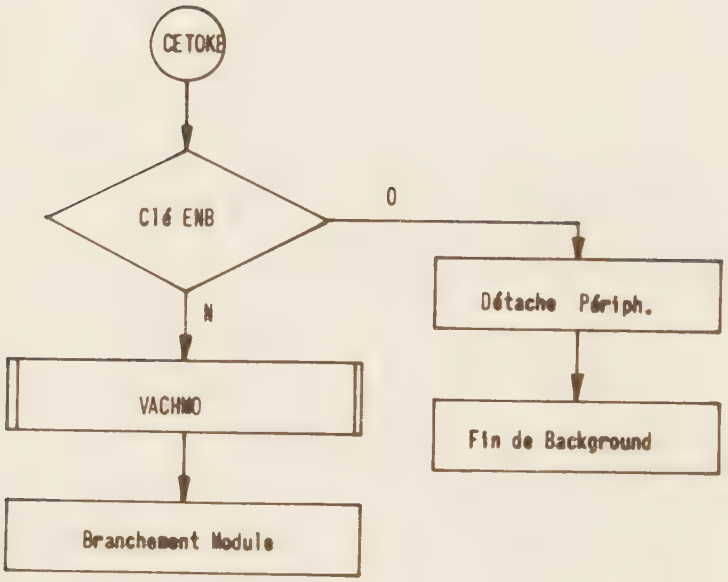


IV. - ORGANIGRAMME DETAILLE




1
MINISU = 1
O
N
Comparaison A avec TACLEB
FORTRA
Comparaison A avec TABCLE
Clé trouvée
N
ERR 4
DEMLIN
Clé trouvée
N
O
ERR 4
X = numéro clé
X = numéro clé
VACECL
DEMLIN
CETOKE
Clé Fichier
O
3 caractères suivants → A
N
A = Z
O
Fichier COR = Fichier SCE
N
2
Rangements 6 caractères suivants → NOMA - NOMB
- 1 → TABFIC
Module présent
O
Impression *
N
Impression *
Branchement Module
VACHMO
DEMLIN

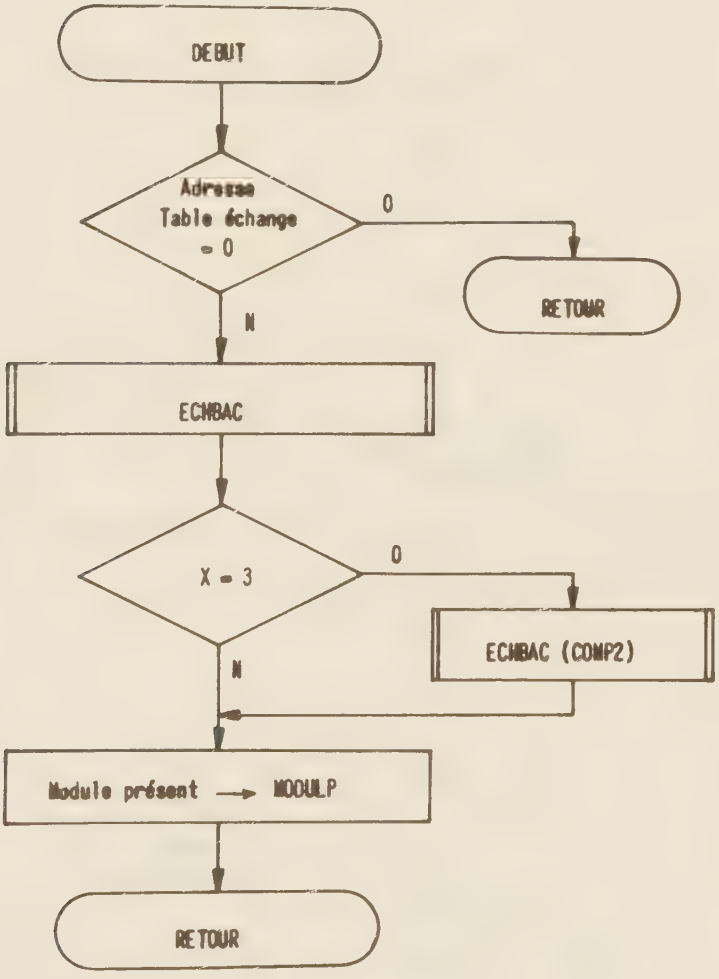

2
Comparaison avec périph.
PERIPH Trouvé
O
N
PERIPH = MEM
N
CEPAME
FICHIER = BUT
N
O
4000→TABFIC
ERR 5
CEPAME
N
PERIPH PAIR
FICHIER IMPAIR
O
Rangement dans TABFIC
IMPRESSION
DEMLIN


CETOKB
Clé ENB
0
N
Détache Périph.
VACHMO
Fin de Background
Branchement Module

SOUS PROGRAMME VACHMO

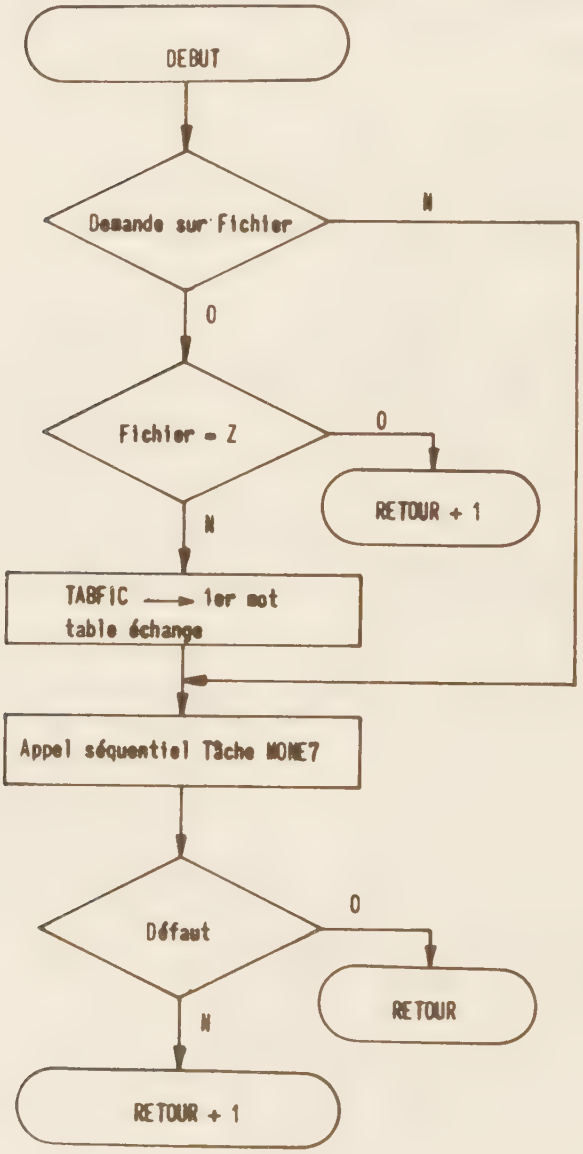
BUT = Swapper in les modules Background appelés et non présents

A l'entrée en a dans X l'index du module monté en mémoire



SOUS PROGRAMME ECHBAC

BUT = Traite toutes demandes à FORES à partir du Background



ANNEXE IV

TABLES UTILISEES

PAR LE SYSTEME GERANT LE CAMAC

TASMG : Table générale contenant les adresses des tables d'USASM de chaque expérience

TASMG

TASAC0	BO ....
TASAC1	BO ...
TASAC2	BO ...
n	⋮ utilisateurs
TASACn	BO....

utilisateur 0 = Système CAMAC

TASAC i : Table des USASM (1 par expérience)

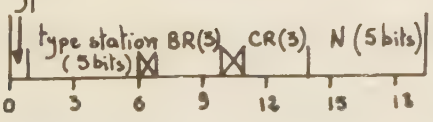
Type 1

Pas d'appels pour cette US (2 mots)

1° Mot : USASM (2 caract. alphanumériques)

2° Mot : Type Station + Adresse

US de type 1

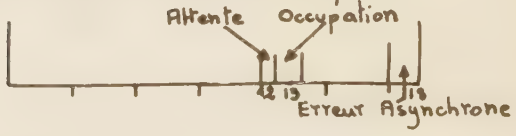


Type 2

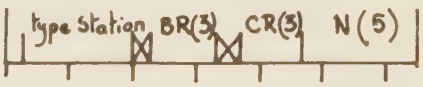
US avec Système d'Appel (4 mots)

1° mot : USASM (2 Caractères)

2° mot : bits caractéristiques



3° Mot : type station + Adresse



4° Mot : adresse de Compte Rendu

Divers types d'Unités Symboliques Fortran (USF)

. Détail du mot 1 de l'USF

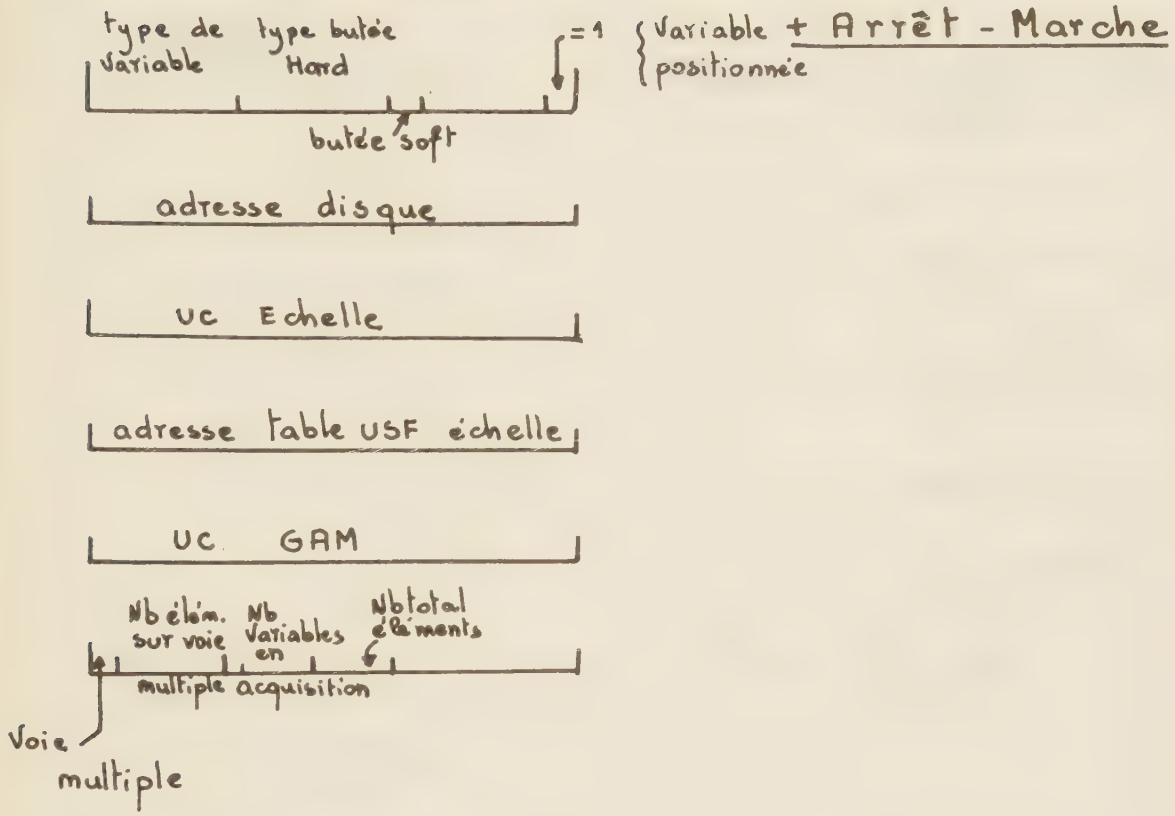
- bits 0-5 : type de variable

.1	Comptage
.2	Moteurs pas à pas + codeur
.3	Moteur continu + codeur Tanzboden + codeurs
.4	Multidétecteur
.5	Moteur pas à pas sans codeur
.6	Moteur de D8
.7	Flipping amplitude
.8	Flipping chronométrie

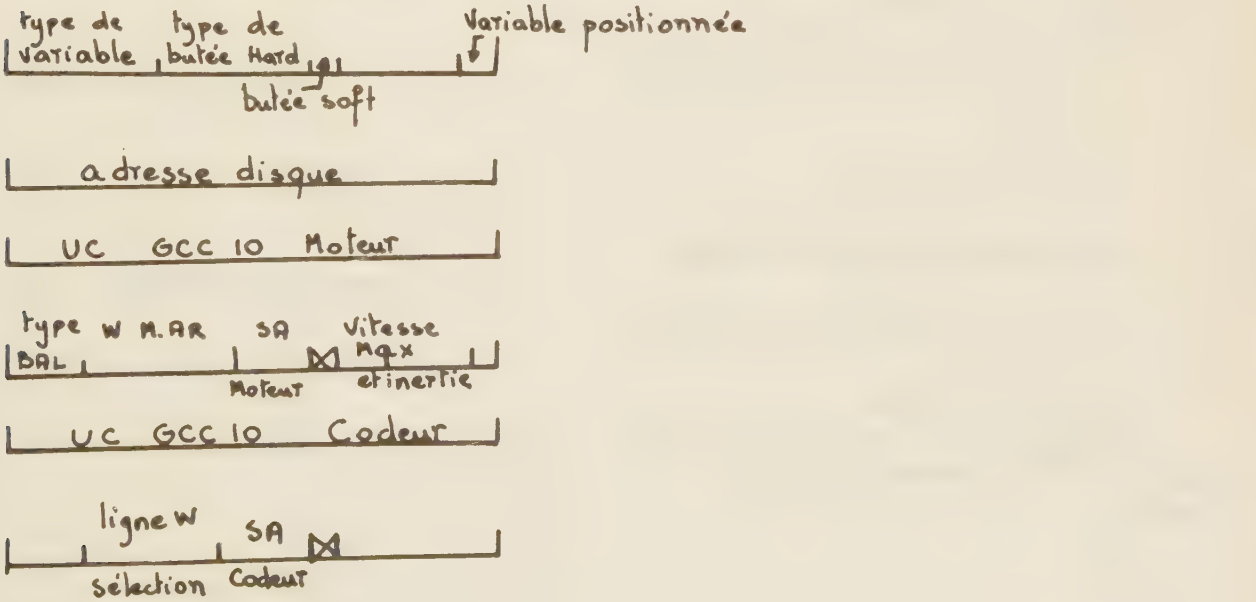
- bits 6-11 : type de butée Hardware

,1	$1 - 10^6$	
,2	0 - 360	
,3	-180	+180
,4	-170	+190
,5		
,6	0	2000

USF de type 1 — Comptage Echelle

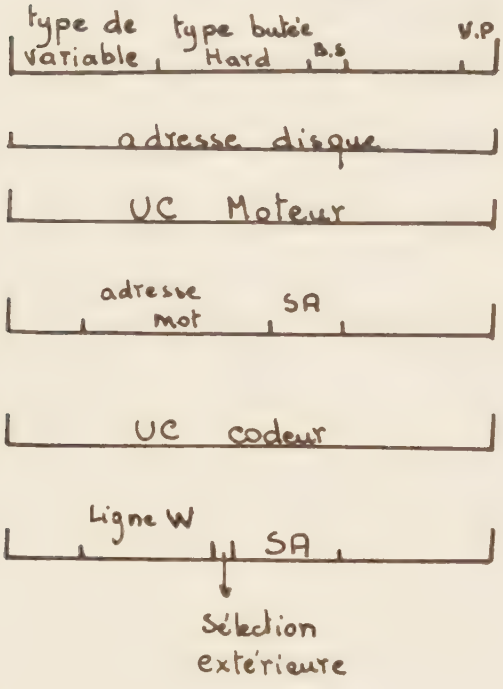


USF de type 2 — Moteurs pas à pas + codeurs



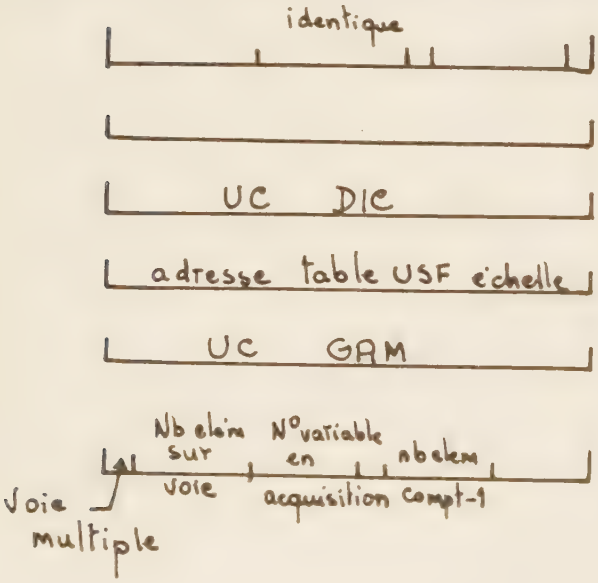
USF de type 3

Moteur Continu - Tanzbotten
-
+ Codeurs

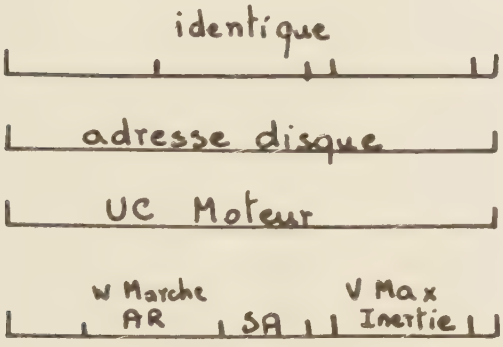


USF de type 4

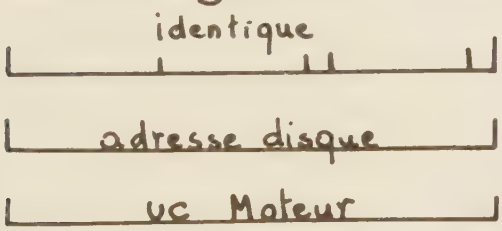
Multidétecteur



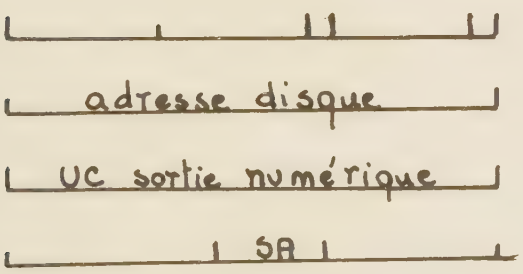
USF de type 5 — Moteur pas à pas sans codeur



USF de type 6 — Moteur D8

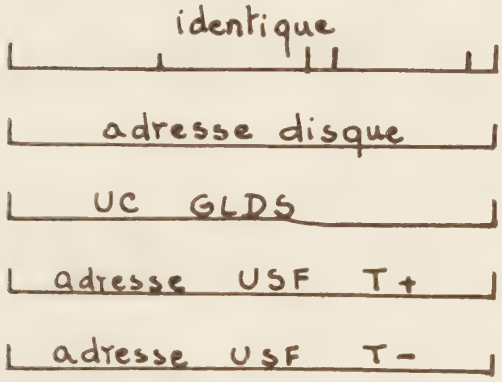


USF de type 7 — Flipping Amplitude

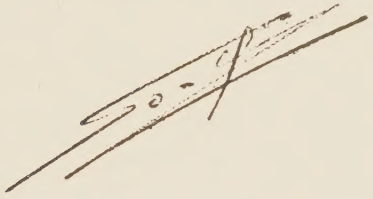


USF de type 8

Flipping chrono



VU, et permis d'imprimer,

Grenoble, le

le Président de l'Université
Scientifique et Médicale